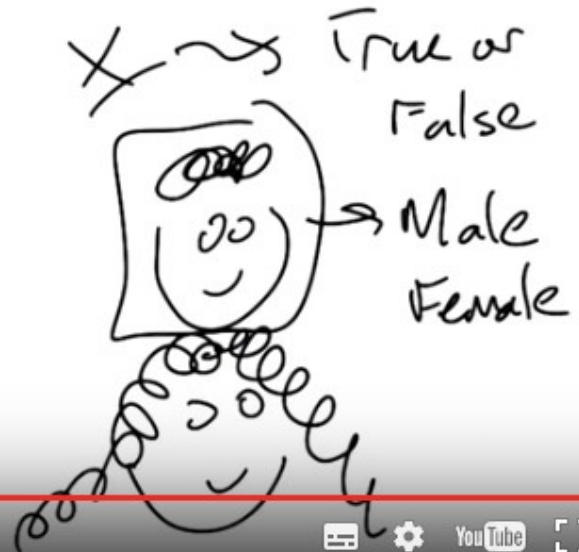
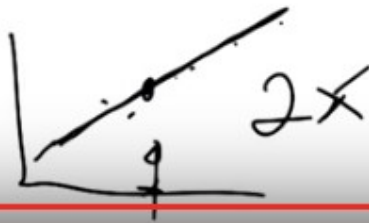


SUPERVISED LEARNING

✓ CLASSIFICATION
✓ REGRESSION





1. Okay, hi Michael. Hey Charles, how's it going. It's going pretty well, how's it going in your end of the world? Very nice, what are we want to talk about today? Well today we are going to talk about supervised learning. But, in particular what we're going to talk about are two kinds of supervised learning, and one particular way to do supervised learning. Okay, so the two types of supervised learning that we typically think about are classification and regression. And we're going to spend most of the time today talking about classification and more time next time talking about regression. So the difference between classification and regression is fairly simple for the purposes of this discussion. Classification is simply the process of taking some kind of input, let's call it x . And I'm going to define these terms in a couple of minutes. And mapping it to some discrete label. Usually, for what we're talking about, something like, true or false. So, what's a good example of that? Imagine that I have a nice little picture of Michael. It looks just like me! It looks exactly like you. So I have a nice little picture here and I want to know whether this is a male. Or a female. So given an input like this I will map it to male or female. So what do you think, Michael? Do you think this is a male or a female? So you're, you're classifying me as male or female based on the picture of me and I would think you know, based on how I look I'm clearly male. Yes. In fact, manly male. So, this would be a classification from pictures to male. The alternative would be something like a picture to female, and I'm just going to take a completely stereotypical image of either a female or. I think it's actually, that's actually me when I let my hair go long. Right, so, so which points out that this can be pretty hard. But this is where we're going to spend most of our time talking about it first as a classification task. So taking some kind of input, in this case pictures, and mapping it to some discrete number of labels, true or false, male or female, car versus cougar(美洲獅), anything that, that you might imagine thinking of. Car versus cougar? Yes. That, I guess that's an important thing if you're driving. You don't want to run into any cougars or probably other cars either. Well you know, you're sitting down and you're trying to decide whether you should ride this thing that you see or not. And if its a cougar maybe you don't want to and if it's a car maybe you do. Excellent. Don't drive a cougar. Don't drive a cougar. That's the first lesson in machine learning. Excellent. Okay, so that's classification. We'll return to regression in a little bit

later during this conversation. But, just as a preview, regression is more about continuous value function. So, something like giving a bunch of points. I want to give in a new point. I want to map it to some real value. So we may pretend that these are examples of a line and so given a point here, I might say the output is right there. Okay, so that's regression but we'll talk about that in a moment. Right now, what I want to talk about is classification. Would an example of regression also be, for example, mapping the pictures of me to the length of my hair? Like a number that represents the length of my hair? Absolutely, for the purposes of, of the sort of things that we're going to be worried about you can really think of the difference between classification and regression is the difference between mapping from some input to some small number of discrete values which might represent concepts. And regression is mapping from some input space to some real number. Potentially infinite number of real numbers. Cool, let's do a, let's do a quiz. Make sure we get this. Okay, I like that.

2. So we've talked about uh, classification and regression and gave a couple of examples of each. So now I want you to take a moment to make certain that you understand the difference because it's a crucial difference for supervised learning. Here's a very short little quiz. You have three questions here and we divided the world up into some input to some learning algorithm, whatever that learning algorithm is. The output that we're expecting and then a box for you to tell us whether its classification or regression. So, the first question, the input is credit history, whatever that means, the number of loans that you have, how much money you make, how many times you've defaulted, how many times you've been late, the sort of things that make up credit history, and the output of the learning algorithm is rather you should lend money or not. So you're a bank, and you're trying to determine whether given a credit history, I should lend this person money, that's question one. Is that classification, or is that regression? Question two you take as input a picture like the examples that we've given before. And the output of your learning system is going to be whether this person is of high school age, college age, or grad student age. The third question is very similar. The input is again a picture. And the output is, I guess, of the actual age of the person, 17, 24, 23 and a half, whatever. So take a moment and try to decide whether these are classification tasks or regression tasks.

SUPERVISED LEARNING

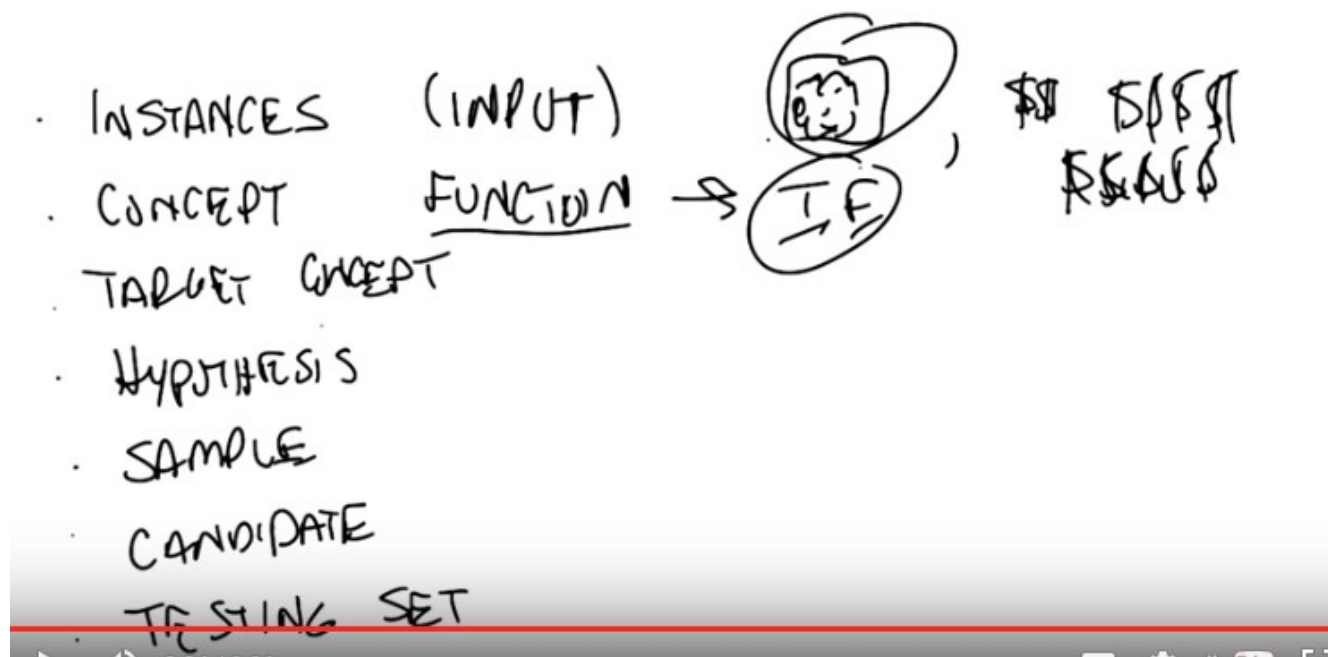
	CLASSIFICATION	OR	REGRESSION	
<u>Input</u> , <u>OUTPUT</u> ,	<u>C</u>	or	<u>R</u>	
1. CREDIT HISTORY , LEND MONEY?	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
2.  , HIGH SCHOOL, COLLEGE, GRAD	<input checked="" type="checkbox"/>		<input type="checkbox"/>	DISCRETE
3.  , AGE	<input type="checkbox"/>		<input checked="" type="checkbox"/>	CONTINUOUS

3. And so now let's give the explanation for the quiz. Alright, so, let's see what happened here. So, what you're saying is in some cases the inputs are discrete or continuous or complicated. In other cases the outputs could be discrete or continuous or complicated. But I think what you were saying is, that what on, on, what matters to determine if something is classification or regression is whether the output is from a discrete small set or, or whether it's some continuous quantity. Is that right? Right, that's exactly right. **The difference between a classification task or a regression task is not about the input, it's about the output. If the output is continuous then it's regression and if the output is small discrete set or discrete set, then it is a classification task.** So, with that in mind, what do you think is the answer to the first one? So, lend money. If it was something like predicting a credit score, that seems like a more continuous quantity. But this is just a yes no, so that's a discrete class, so I'm going to go with classification. That is, correct. It is classification and the short answer is, because it's a binary task. True, false. Yes, no. Lend money or don't lend money. Got it. So it's a simple classification test. Okay, with that in mind, what about number two? Alright, so number two. It's trying to judge something about where they fall on a scale, high school, college, or grad student. But all of, the system is being asked to do is put them into one of those three categories, and these categories are like classes, so it's classification. That is also exactly right. Classification. We moved from binary to trinary in this case, but the important thing is that it's discrete. So it doesn't matter if it's high school, college grad, professor, elementary school, any number of other ways we might decide where your status of matriculation is is a small discrete set. So, with that in mind, **what about number three?** Alright, so the input is the same in this case. And the output is kind of the same except there's, well there's certainly more categories because there's more possible ages than just those three. But **when you gave the example you did explicitly say that ages can be fractional like, you know, 22.3.** So that definitely makes me think that it's continuous, so it should be regression. Right, I think that is exactly the right thing, you have a continuous output.

<u>SUPERVISED LEARNING</u>		CLASSIFICATION or REGRESSION?		
<u>Input</u>	<u>OUTPUT</u>	<u>C</u>	<u>R</u>	
1. CREDIT HISTORY	LEND money?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	BINARY (2)
2. 	HIGH SCHOOL, COLLEGE, GRAD	<input checked="" type="checkbox"/>	<input type="checkbox"/>	DISCRETE (3) (250)
3. 	AGE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CONTINUOUS

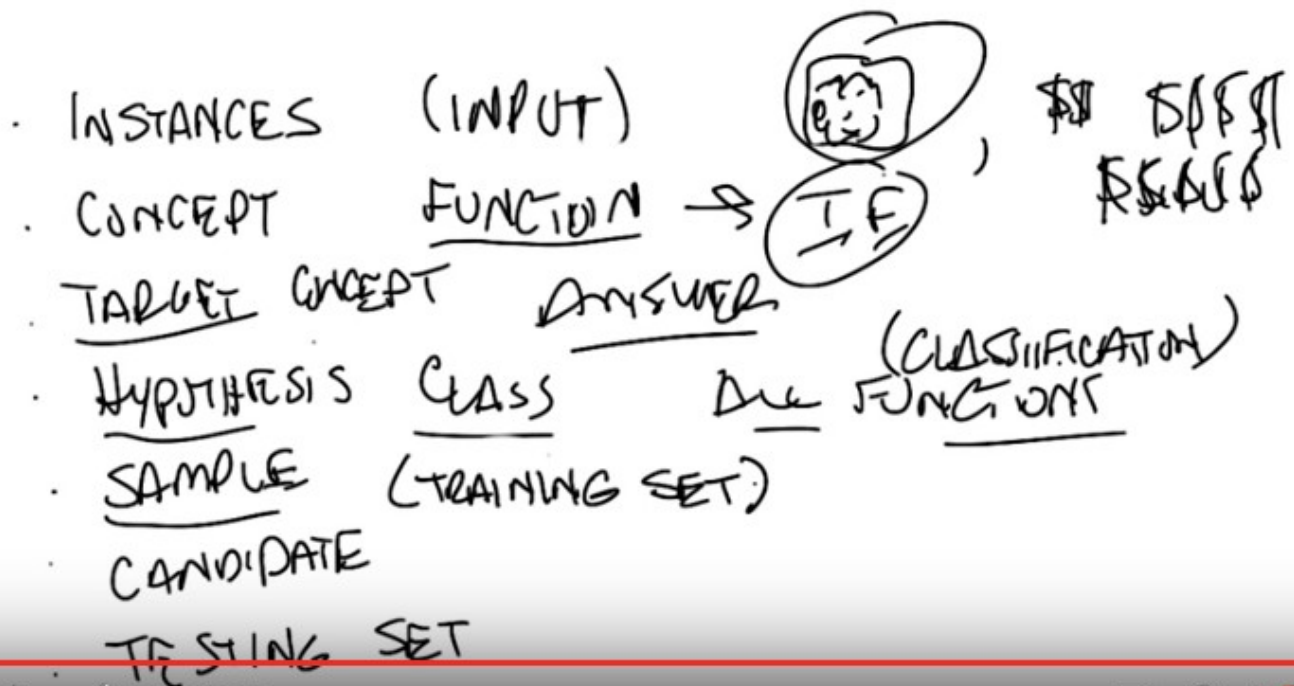
Now, I do want to point something out. That **while the right answer is regression, a lot of people might have decided that the answer was classification.** So, what's an argument? If I told you in fact the answer was classification, what would be your argument for why that would be? Well, I guess. Can you

think of one? Yeah, I guess. I mean, you know, if you think about ages as being discrete. You just say, you know, you can one or two or three or, you know, whatever up to 130, say. But there, but there's just that, that set. There isn't really, you know, usually we don't talk about fractional ages. So, so it seems like you could think of it as, as, as a set of classes. Right. So let's imagine. So, how old are people? **Let's imagine we only cared about years, so you're either one or two or three or four or five.** Or maybe you can be one and a half, and two and a half, and three and a half. But, whatever, it's, it's not all possible real number values. **And we know that people don't live beyond, say, 250. Well, in that case, you've got a very large discrete set but it's still discrete.** Doesn't matter whether there's two things in your set, three things in your set, or in this case 250 things in your set, it's still discrete. So, whether it's regression, or classification, depends upon exactly how you define your output and these things become important. I'm going to argue that in practice, if you were going to set up this problem, the easiest way to do it would be to think about it as a real number and you would predict something like 23.7. And so it's going to end up being a regression task and we can might, maybe think about that a little bit more as we move along. So either answer would be acceptable depending upon what your explanation of exactly what the output was. Was. You buy that? That makes sense. Excellent. Okay. Alright, let's move beyond the quiz and start thinking about exactly what it means to define a classification problem. And then later what it means to define a regression problem.



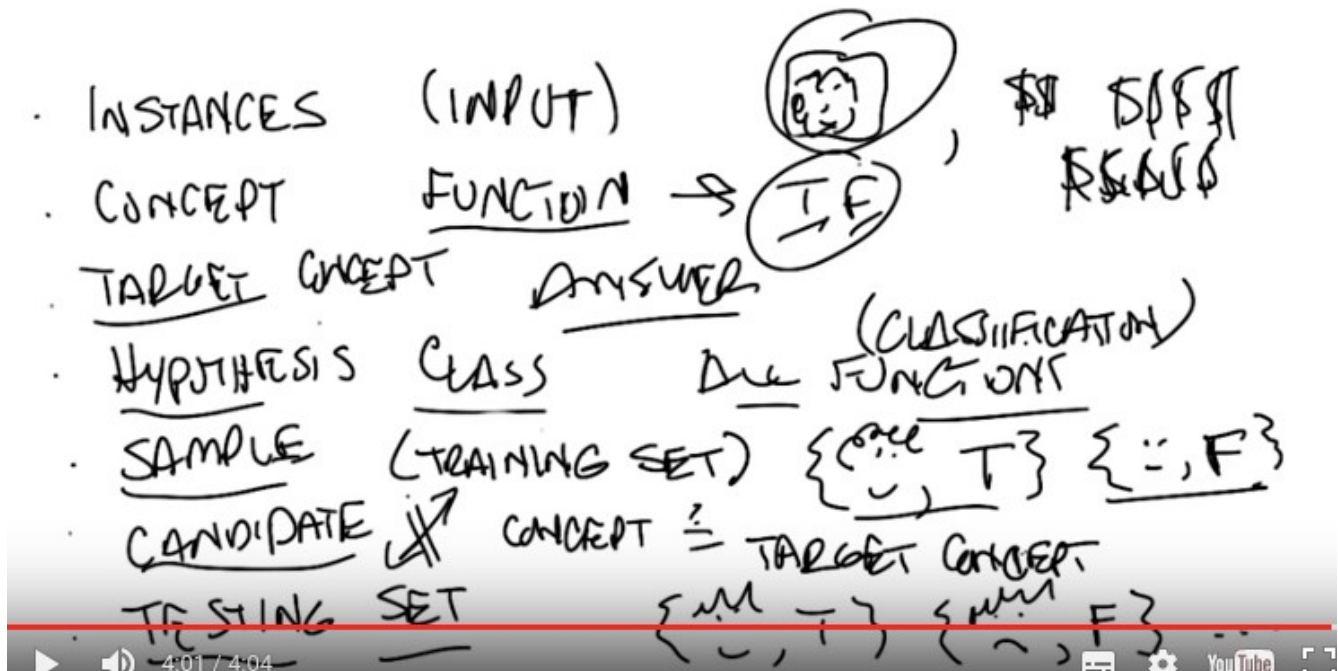
its discrete values, those are your instances, the set of things that you're looking at. So you have instances, that's the set of inputs that you have. And then what we're trying to find is some kind of function and that is the concept that we care about. And this function actually maps inputs to outputs, so it takes the instances, it maps them in this case to some kind of output such as true or false. This is the, the categories of things that we're worried about. And for most of the conversation that we're going to be having, we're going to be thinking about binary classification, just true and false. But the truth is, this would work whether there were three outputs, as we did with high school, college, or grad school, or whether there were 250 as we were contemplating for ages. But the main thing here is the concept, is the function that we care about that's going to map pictures, for example, to true or false. So okay, I get, I get the use of the word, "instance", right? "Instance" is just, like, a single thing that's out there. But I have an intuitive notion of what a concept is. How does that relate to this more formal notion? Like, can we connect this to, kind of, the intuitive notion of what a concept is? I guess so. So a concept, I don't know. How would you want to put that? So a concept is something That, I mean were talking about is a notion of a function, so what it means formally is that you have some input, like a picture, and it immediately inputs maps anything in that input space to some particular defined output space, like true or false, or male or female, or credit, credit worthy or not. Intuitively a concept is an idea describes a set of things. OK, so we can talk about maleness or femaleness. We can talk about short and tall; we can talk about college students and grad students. And so the concept is the notion of what creditworthiness is, what a male is, what a college student is. Okay, I think I see that. So, so essentially if you want to think of tallness, the concept of tallness, one way to define it is to say, Well in general if you give me something I can tell you rather or not its tall, so it's going to map those somethings to am I tall or not. True or false. Right. Exactly and so really when you think about any concept and we talk about this in generally airai~ a concept is effectively a way of saying is effectively a set of things that are apart of that concept. So, you can have a concept of a car and if I gave you "cars" you would say these things are in it and if I gave you "non-cars" you would say they are not. And so a concept is, therefore by definition, a mapping between objects in a world and membership in a set, which makes it a function. Okay, that makes sense. Okay.

CLASSIFICATION LEARNING



5. So with that, with that notion of a concept as functions or as mappings from objects to membership in a set we have a particular target concept. And the only difference between a target concept and the general notion of concept is that the target concept is the thing we're trying to find. It's the actual answer. All right? So, a function that determines whether something is a car or not, or male or not, is the target concept. Now this is actually important, right, because we could say that. We have this notion in our head of things that are cars or things that are males, or thing, or people who are credit worthy but unless we actually have it written down somewhere we don't know whether it's right or wrong. So there is some target concept we're trying to get of all the concepts that might map pictures or people to true and false. Okay, so if you trying to teach me what tallness is so you have this kind of concept in mind of these, these things are tall and these things are not tall. So you're going to have to somehow convey that to me. So how are you going to teach me? So that's exactly the, well that's what comes up with the rest of these things that we're defining here. Okay. So let me tell you what the next four things are then you can tell me whether that answers your question. Got it. How about that? OK, so we've got a notion of instances, input, we've got a notion of concepts. Which take input and maps into some kind of output. And we've got there's some target constant, some specific function, some particular idea that we're trying to find, we're trying to represent. But out of what? So that's where the hypothesis comes in. And in fact I think it's better to say hypothesis class. So that's the set of all concepts that you're willing to entertain(考慮). So it's all the functions I'm willing to think about. So why wouldn't it just be all possible functions? It could be all possible functions and that's a perfectly reasonable hypothesis class. The problem with that is that if it is all possible functions it may be very, very hard for you. Figure out which function is the right one given finite data. So when we actually go over decision trees next I think it will be kind of clear why you need to pick a specific hypothesis class. Okay. So let's return to that in a little bit but it's an excellent question. So, conceptually in the back of your head until we, we come to specific examples, you can think of hypothesis class as simply being all possible functions in the world. On the other hand, even so far just the classification learning, we already know that we're restricting ourselves to a subset of all possible functions in the world, because all possible functions in the world includes things like x squared, and that's regression. And we've already said, we're not doing regression. We're doing classification. So already hypothesis classes all functions we care about and maybe it's all classification functions. So we've already picked a subset. So we got all these incidences, got all these concepts, we want to find a, a particular concept and we've got this set of functions we're willing to look at. So how are we going to determine what the right answer is. So if you try to answer Micheal's question that we do that in machine learning is with a sample or another name for which I prefer is a training set.

CLASSIFICATION LEARNING



MO: training set is used to learn or draw conclusions, testing set is used to test whether our conclusions are right.

From online:

++Training set (60% of the original data set): This is used to build up our prediction algorithm. Our algorithm tries to tune itself to the quirks of the training data sets. In this phase we usually create multiple algorithms in order to compare their performances during the Cross-Validation Phase.

++Cross-Validation set (20% of the original data set): This data set is used to compare the performances of the prediction algorithms that were created based on the training set. We choose the algorithm that has the best performance.

From another site: A validation set, that we use to compare the performance of our model, using different parameters for our algorithm of choice. We then select the parameters that maximize our accuracy. This is usually 10 to 20 percent of the dataset.

Tao: cross-validation set 的作用就是: 將 train set 弄出來的 decision tree 拿來跑 cross-validation set, 看其準確率, 然後調節參數, 使其在 validation-set 中的準確率最大. 此時有人要問: train set 不是已經將 decision tree 給弄好了, 參數也選好了嗎, 為何還要用 validation-set 來調參數? 答: train set 中的準確率並不可信, 因為就是用它們來 train 的(這也是為何要用 test set 來測準確率), 而我們在此時還沒到用 test set 的階段(test 過程一般不再改參數了), 所以只能用 cross-validation set 來估計準確率, 然後調參數. 而這種做法也跟後面講的 cross validation 方法是一致的, 因為每次用不同的 20% 的 original data set 來做 cross-validation set, 就跟後面講的 cross validation 方法一致了.

++Test set (20% of the original data set): Now we have chosen our preferred prediction algorithm but we don't know yet how it's going to perform on completely unseen real-world data. So, we apply our

chosen prediction algorithm on our test set in order to see how it's going to perform so we can have an idea about our algorithm's performance on unseen data.

From online:

Last year, I followed Prof: Andrew Ng's online machine learning course. His recommendation was

Training: 60%

Cross validation: 20%

Testing: 20%

Tao:

Training 的時候是知道結果的, 即知道某一個數據點是被 classify 為哪一類. Training 的過程就是用這些數據點 及其結果, 來求出算法的參數值(如 decision tree 中 root 節點是哪個 feature, neural network 中的 w_i 值, 等等), 這就得到了一個確定的算法(或步驟), 然後再用這個算法 來算一遍 training data 中每個數據點的結果, 其正確率即為 training 的正確率(此正確率經常不是 100%, think of fitting). 然後再用這個算法 來算一遍 testing data 中每個數據點的結果, 其正確率即為 testing 的正確率, testing 的正確率 才是我們真正關心的東西 (即 testing 的正確率 是用來衡量一個算法好壞的). 最後再將此算法用於 實際數據 中去 practice.

6. So what's a training set? Well a training set is a set of all of our inputs, like pictures of people, paired with a label, which is the correct output. So in this case, yes, this person is credit worthy. [LAUGH] Versus another example. You can tell I'm credit worthy based on my curly hair. Purely on the hair. Versus someone who has no curly hair and therefore is obviously not credit worthy. And if you get bunches and bunches of examples of input and output pairs, that's a training set. And that's what's going to be the basis for you figuring out what is the correct concept or function. I see. So instead of just telling me what tall means, you're going to give me lots of examples of, this is tall, this is not tall, this is tall, this is tall, this is tall, this is not tall. And that's the way that you're explaining what the target concept is. Right. So if you want to think about this in the real world, it's as if we're walking down the street and I'm pointing out cars to you, and non-cars to you, rather than trying to give you a dictionary that defines exactly what a car is. And that is fundamentally inductive learning as we talked about before. Lots and lots of examples, lots of labels. Now I have to generalize beyond that. So, last few things that we we talk about, last two terms I want to introduce are candidate, and testing set. So what's a candidate? Well a candidate is just simply the, a concept that you think might be the target concept. So, for example, I might have, right now, you already did this where you said, oh, okay I see, clearly I'm credit worthy because I have curly hair. So, you've effectively asserted a particular function that looks at, looks for curly hair, and says, if there's curly hair there, the person's credit worthy. Which is certainly how I think about it. And people who are not curly hair, or do not have curly hair are, in fact, not credit worthy. So, that's your target concept. And so, then, the question is, given that you have a bunch of examples, and you have a particular candidate or a candidate concept, how do you know whether you are right or wrong? How do you know whether it's a good candidate or not? And that's where the testing set comes in. So a testing set looks just like a training set. So here our training set, we'll have pictures and whether someone turns out to be credit worthy or not. And I will take your candidate concept and I'll determine whether it does a good job or not, by looking at the testing set. So in this case, because you decided curly hair matters, I have drawn, I have given you two examples from a training set, both of which have curly hair, but only one of which is deemed credit worthy. Which means your target concept is probably not right. So to do that test I, guess you can go through all the pictures in the testing set, apply the candidate concept to see whether it says true or false, and then compare that to what the testing set actually says that answer is. Right, and that'll give you an error. So, by the way, the true target, the true target concept is whether you smile or not. Oh. That does make somebody credit worthy. It does in my world. Or at least I, wish it did in my world. Okay. So, by the way an important point is that the training set and the testing set should not be the same. If you learn

from your training set, and I test you only on your training set, then that's considered cheating in the machine learning world. Because then you haven't really shown the ability to generalize. So typically we want the training set to include lots of examples that you don't, the testing set, I'm sorry, to include lots of examples that you don't see in your training set. And that is proof that you're able to generalize. I see. So that, and that makes intuitive sense, right? So, like, if, if you're a teacher and you're telling me, you give me a bunch of fact and then you test me exactly that bunch of facts, it doesn't, I don't have to have understood them. I just can regurgitate them back. If you really want to see if I got the right concept, you have to see whether or not I can apply that concept in new examples. Yes, which is exactly why our final exams are written the way that they are written. Because you can argue that I've learned something by doing memorization, but the truth is you haven't. You've just memorized. Here you have to do generalization. As you remember from our last discussion, **generalization is the whole point of machine learning.**

DECISION TREES



ENTER?

DECISION TREES

EATERY:

~~TYPE~~ ITALIAN, FRENCH,
THAI
~~ATMOS~~ FANCY, HIW
CASUAL
OCCUPIED? ✓ X
HOT DATE? ✓ X

HIW: hole-in-the-wall(老師隨便編的縮寫, 不重要. hole-in-the-wall 應該是 fancy 的反義詞, 即 the building 只是 a hole in the wall, nothing fancy)

7. All right, so we've defined our terms, we know, what it takes to do, at least supervised learning. So now I want to do a specific algorithm and a specific representation, that allows us to solve the problem of going from instances to, actual concepts. So what we're going to talk about next are decision trees. And I think the best way to introduce decision trees is through an example. So, here's the very simple example I want you to think about for a while. You're on a date with someone. And you come to a restaurant. And you're going to try to decide whether to enter the restaurant or not. So, your, input, your instances are going to be features about the restaurant. And we'll talk a little bit about what those features might be. And the output is whether you should enter or not. Okay, so it's a very simple, straightforward problem but there are a lot of details here that we have to figure out. It's a classification problem. It's clearly a classification problem because the output is yes, we should enter or no, we should move on to the next restaurant. So in fact, it's not just a classification problem, it's those binary classification problems that I said that we'd almost exclusively be thinking about for the next few minutes. Okay. So, you understand the problem set up? Yes, though I'm not sure exactly what the pieces of the input are. Right, so that's actually the right next question to ask. We have to actually be specific now about a representation. Before I was drawing squiggly little lines and you could imagine what they were, but now since we're really going to go through an example, we need to be clear about what it means to be standing in front of the restaurant. So, let's try to figure out how we would represent that, how we would define that. We're talking about, you're standing in front of a restaurant or eatery(餐館) because I can't see the reliably small restaurant. And we're going to try to figure out whether we're going to go in or not. But, what do we have to describe our eatery? What do we have? What are our attributes? What are the instances actually made of? So, what in, or another way of putting it is, what are the features that we need to pay attention to that are going to help us to determine whether we should yes, enter into the restaurant. Or no, move on to the next restaurant. So,

any ideas Michael? Sure. I guess there's like the type of restaurant. Okay, Oh, is it tall or short, and is it a good credit risk? [LAUGH] Oh wait, no, no, no wait, I know. Like the Italian versus French, versus, you know, Vietnamese. So let's call that the type. So it could be Italian, it could be French, it could be Thai, it could be American, there are American restaurants, right? Sure. Greek, it can be, Armenian. It can any kind of restaurant you want to. Okay, good. So that's something that probably matters because maybe you don't like Italian food or maybe you're really in the mood for Thai. Sounds perfect. Okay anything else? Sure. How about whether it smells good? You know, I like cleanliness. Let's let's, or you know what, let's, let's be nice to our eateries and let's say atmosphere. Mm. Right because if there's, you know, no atmosphere, then it is going to be really hard to breathe. That's exactly right. So is it fancy? Is it a hole-in-the-wall, which I'm going to spell HIW. Is it a hole-in-the-wall, umm, those sorts of things. The, you know? Casual, I guess, is another category. Casual. And so on, and so forth. You could imagine lots of things like that, but these things might matter to you and your date. Okay, so, we know the type of the restaurant that we have, we know whether it's fancy, whether it's casual, whether it's a hole in the wall. Some of the best food I've ever had are in you know, well-known hole in the walls. Those sorts of things. Anything else you can think of? Sure, Sometimes, I might use something like looking inside and seeing whether there's people in there and whether they look they're having a good time. Right. So that's an important thing. So let's just say If it's occupied. Now why might that matter in reality? Well it matters because if it's completely full and you may have to wait for a very long time, you might not want to go in. On the other hand. If you're looking at a restaurant you've never heard of, and there's only two people in it, and it's Friday at 7 p.m. Maybe that says something about something. Maybe you want it to be quiet. You know, those sorts of things might matter. Okay, so, we've got type, we've got atmosphere, we've got occupied. Anything else you can think of? And I have been out of the dating market for a while, but I guess it could imagine, I could imagine how hard I am trying to work to impress my date. perfect. So do you have a hot date or not? Or, this is someone who you really, really, really want to impress and so, maybe it matters then, it's even more important whether it's a fancy restaurant or a hole in the wall, or whether it's French or whether it's an American restaurant. That make sense? I think that makes sense. Notice, by the way, that the first two sets that we have have multiple possible categories here. So it could be Italian, French, Thai, American, so on and so forth. Atmosphere is something that can have many, many possible values, but the last two things that we talked about were all binary. Either it's occupied or it's not. Yes or no or, you have a hot date or you don't. And I think we could go on like this for a long time but, let's try to move on to maybe a couple of other features and then try to actually figure out how we may actually solve this.

DECISION TREES

EATERY:

COST \$, \$\$, \$\$\$
 \$
 #

HUNGRY?

DRAINING?

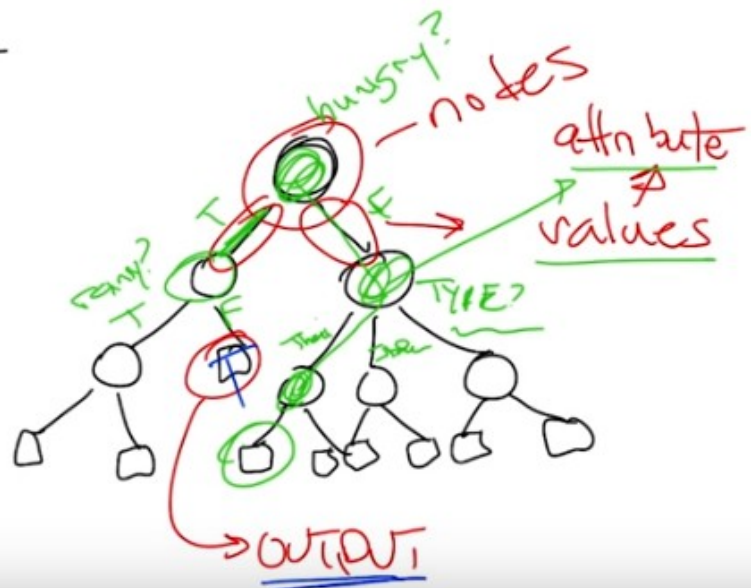
8. Alright, so Michael. Last set of features that that's come up with three or four, three or four more features and then move on. Sure. So come up with a couple. Alright, so I could, sometimes I'll look at the menu that's posted outside, and I'll see if the, you know? How pricey it is. Okay, so cost. Right, so cost can be represented as discrete input. By the way, it could also be represented as an actual number. Right? We could say, look it's cheap, it's moderately expensive, it's really expensive or you could have a number which is the average cost of an entry. And it doesn't really matter for, for what we're talking about now but just some way of representing cost. Okay. Just give me one or two more features but I want to give me some features that don't have anything to do with the restaurant itself but might still be important. Hmm. because, by the way, hot date probably doesn't have anything to do with the restaurant itself. So, even though we've been talking the features of the restaurant. We've actually been picking up, at least, one feature that doesn't have anything to do directly with the restaurant itself. Not to. So, whether I'm hungry? I like that. Here's another one. What's the weather like. Is it raining outside? Which is a different sense of atmosphere. Right. because if it's raining outside, maybe it's not your, your favorite choice but you don't want to walk anymore. Okay, so we have a ton of features here. We've gone through a few of them. Notice that some of the specifically have to do with the restaurant and some of them have to do with things that are external to the restaurant itself but you can imagine that they're all important. Or possibly important to whether you should enter into the restaurant or not. Agreed? And there's a bunch of features you could imagine coming up with that probably have nothing to do with whether you should enter into the restaurant or not. Like, how many cars are currently parked across the country. Probably doesn't have an impact on whether you're going to enter into a specific restaurant or not. Okay. So, we have a whole bunch of features and right now we're sticking with features we think that might be relevant. And we're going to use those to make some kind of decision.

DECISION TREES

REPRESENTATION

VS

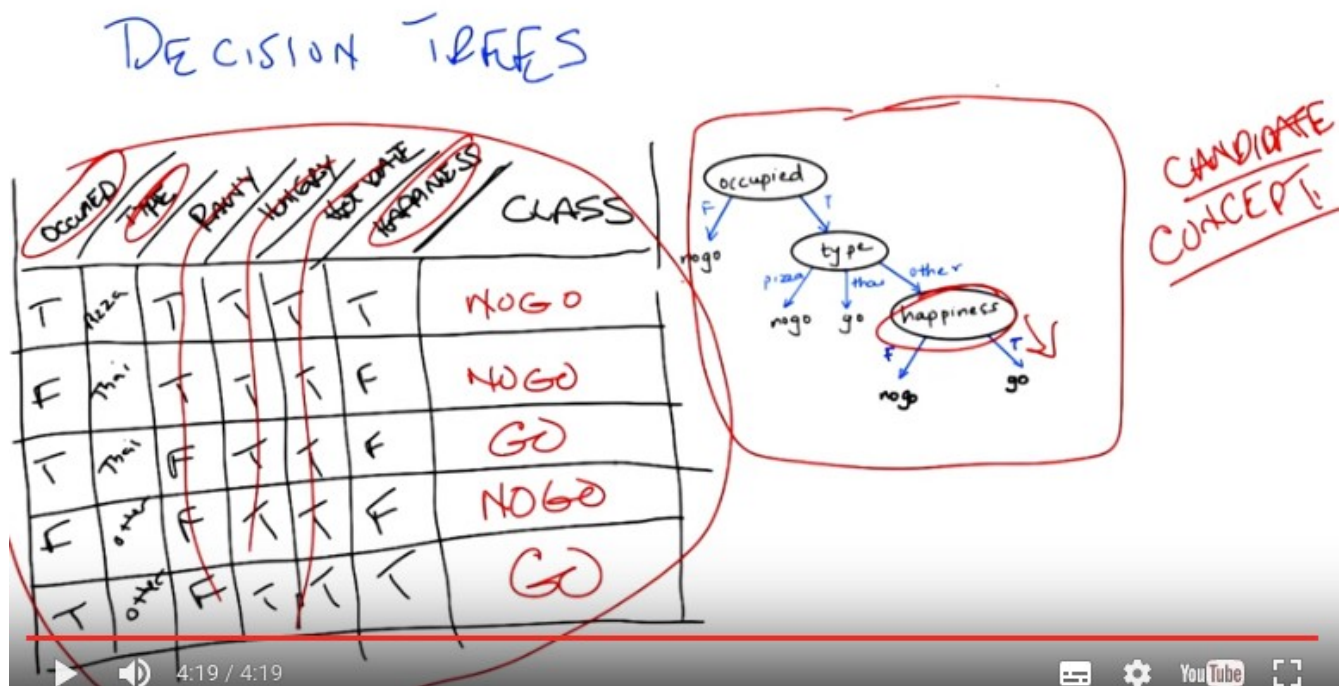
ALGORITHM



So, that gets us to decision trees. So, the first thing, that, that we want to do is, we want to separate out what a decision tree is from the algorithm that you might use to build the decision tree. So the first thing we want to think about is the fact that a decision tree has a specific representation. Then only after we understand that representation and go through an example, we'll start thinking about an algorithm for finding or building a decision tree. Okay, are you with me? Yeah. Excellent. Okay, so a decision tree is a very simple thing. Well, you might be, might be surprised to know it's a tree, the first part of it. And it looks kind of like this. So, what I've drawn for you is example. Sample generic, decision tree. And what you'll see is three parts to it. The first thing you'll see is a circle. These are called nodes, and these are in fact, decision nodes. So, what you do here, is you pick a particular attribute. And you ask a question about it. The answer to that question, which is its value for what the edges represent in your tree. Okay. So we have these nodes which represent attributes, and we have edges which represent value. So let's be specific about what that means. So here's a particular attribute we might pick for the top node here. Let's call it hungry. That's one of the features that Michael came up with. Am I hungry or not? And there's only two possible answers for that. yes, I'm hungry, true, or false, I am not hungry. And each of these nodes represent some attribute. And the edges represent the answers for specific values. So it's as if I'm making a bunch of decisions by asking a series of questions. Am I hungry? And if the answer is yes, I am hungry, then I go and I ask a different question. Like is it rainy outside? And maybe it is rainy and maybe it's not rainy, and let's say if it isn't rainy then I want to make a decision, and so these square boxes here are the actual output. Okay so it's hungry, so you're hungry, yes, and it's not raining, so what do you do? So, let's just say you go in. True, I go in so, when it's, I'm hungry and it's not raining, I go in. And that, that true is answering a different question. It's not in the nodes I guess. So, in the leaves, the t and f means something different. That's right. It's the out, that's exactly right. The, the leaves, the little boxes, the leaves of your decision tree is your answer. What's on the on the edges are the possible values that your attribute can take on. So, in fact, let's try to, let's make that clear by picking a different by picking another possible attribute. You could imagine that if I am not hungry, what's going to matter a lot now is say, the type of restaurant, right. Which we said there were many, many types of restaurants. So, you know thai, I forget [CROSSTALK] Italian. Italian, and you know, something. French fries. And French Fries. So if I'm not hungry, then what matters a lot more is the type of restaurant, and so I'll move down this path instead and start

asking other questions. But ultimately, no matter how I, what this decision tree allows me to do is to ask a series of questions and depending upon those answers, move down the tree, until eventually I have some particular output answer, yes I go in the restaurant, or no I do not. Ok this is still seeming a little abstract to me, can we, can we maybe work through a very concrete example. Yeah, I think that makes a lot of sense. Let's do a quiz. Ha, okay, let's do a quiz.

9. Okay, so we've now seen an abstract example of decision trees. So let's make certain that we understand it with a concrete example. So, to your right is a specific decision tree that looks at some of the features that we talked about. Whether you are occupied or not, whether the restaurant is occupied or not what type of restaurant it is. Your choices are pizza, Thai, or other. And whether the people inside look happy or not. The possible outputs are again binary either you don't go or you do go, into the restaurant. On your left is a table which has six of the features that we've talked about. Whether the restaurant is occupied or not, the type of restaurant, whether it's raining outside or not, whether your hungry or not. Whether you're on a hot and whether the people inside look happy, and some values for each of those. And what we would like for you to do is to tell us what the output of this decision tree would be in each case. Alright, so the decision tree here is a, it's a classifier, but we had another name. Oh, it's a concept. Yes. Alright, and each row of this is a different time that we're stopping at a restaurant, and the, the little values there summarize what is true about this particular situation. And, and you're saying we need to then trace through this decision tree and figure out what class is, okay yeah, that's what you said. Okay, I'm ready. Alright, perfect. Okay, that's the quiz, go.



10. You've got your answers. Let me tell you what we think the answers are. Now the nice thing about a decision tree sort of conceptually and intuitively, is that it really is asking a series of questions. So, we can simply look at these rows over here and the values that our features have and we can just follow down the path of the tree. So, in the first case. We have true. We have true for occupied, which means we want to go down the right side of the tree. And check on the type. So in the first case, the type is pizza. And so we go down the first branch and that means. We do not go down the tree. So, the output is no go. So, okay, so now, I got a different answer. So, I looked at this and I said happiness is true.

And, the bottom box says happiness true, you go. Right. So, you got the wrong, you got what I'm going to tell you is the wrong answer by going from the bottom of the tree up. The way decision trees work is you always start at the root of the tree. That is the very top of the tree. And you ask the questions in that order. It just seems like it would be faster to start at the bottom. Yeah but then you would never look at anything else in the tree. Good point. All right so. If you start at the bottom, you can't go up. Alright. So, okay, so let me see if I get this straight. So I'll, I'll do the, the second instance. The second instance, you say that we need to start at the top of the tree where it says occupied. And so now I look at the instance and the instance says that it's false for occupied, so we go down that left branch and we hit no go. Oh wait but now I haven't look at any of the other nodes. You don't have to look at any of the other nodes because it turns out that if it's not occupied you just don't go into a restaurant. So you're the type of person who doesn't like to be the only person in a restaurant. Got it, all right, so that's a no go. So that's a no-go. That's an important point, Michael. Actually, you might also notice that this whole tree, even if you look at every single feature in the tree, only has three of the attributes. It only looks at occupied. Type. And happiness. I see. So hot date is sort of irrelevant which is good, because in this case it's not really changing from instance to instance anyway. True. And neither is hungry you might notice. Oh, I am kind of hungry. Although, I'm always hungry. Although raining does in fact change a little bit here and there. But it apparently it doesn't matter. I see. Because you always take an umbrella. Got it. Okay, so let's quickly do the other three and see if we come to the same conclusion. Alright. Well all the instances that have occupied false we know those are no go, right away. Oh, good point. So we can do it kind of out of order. And the other ones are both occupied. One is tie and one is other. For the tie one we go. The other one, oh I see, for the other one we have to look at whether there's happiness or not, and in this instance happiness is true. So we get on the right branch and we go. And we go. Exactly it. So we notice hot date doesn't matter, hungry doesn't matter and rainy doesn't matter. And the only thing that matters are whether you're occupied, what type of restaurant you're at and whether you're happy or not. Or whether the, the patrons in the restaurants are restaurant is, are happy or not. But, here's the other thing about this. It's not just about the features. Let's tie it back in to the other things, that we mentioned in the beginning. This, in our case, [this table actually represents our testing set. It's the thing that we're going to look at to determine whether we were right or wrong. These are the examples that we're going to do to see whether we generalize or not. And this particular tree here is our candidate concept. So there's lots and lots and lots of other trees that we might have used. We might have used a tree that also took, asked questions about whether it was rainy or not or asked questions about whether you were on a hot date or not. But we didn't. We picked a specific tree that had only these three features and only asked in this particular way. So what we're going to talk about next. Is how we might decide whether to choose this tree over any of the other possible number of trees that we might have chosen instead.](#)

20 QUESTIONS

FURTHER
NARROW
POSSIBILITIES

- ANIMAL? ✓
- PERSON? ✓
- FAMOUS? ✓
- WE KNOW PERSONALLY? X
- LIVING? X
- MUSIC? ✓
- 20th century? ✓
- RAP? X
- SINGER? ✓
- FEMALE? X
- RECENT DEATH? ✓
- MICHAEL? ✓
- JACKSON? ✓

11. Alright, so we understand at this point how to use these decision trees, but this is a class about machine learning, right? So we need to figure out how to make those decision trees happen as a result of processing a set of training data. So, it's not clear to me how we're going to make that leap. Let's see if we can figure it out together. So, I'm going to play a game with you, Michael, and if we do the game well. Then I think we'll have an idea of what a good algorithm might be for actually building a decision tree. So we're going to play 20 questions. You're familiar with 20 questions? Right, that's the game where I'm allowed to ask yes/no questions to try to guess something that you're thinking of, and if I take more than 20 of them, then I lose. Right, okay. So, here's what I'm going to do. I'm going to think of a thing, and, you ask me questions and I'm going to answer them and we'll see if you can guess what thing I'm thinking about. Okay, I've got something in my head. The first question, the typical first question is it animal, vegetable or mineral but that doesn't seem like a yes/no question, so is it, is it an animal, or like a, a living creature. The answer is yes. Alright, is it a person? Is it a person? The answer is yes. Is it a famous person? Is it a famous person? That's a deep philosophical question, but I'm going to say yes. Is it a famous person that we both know in like directly. Oh, who we both personally know directly, I do not believe so. Yeah. So, the answer is no. Is it a living person? Living person, no. So it is a dead, famous person. Is the person famous for, say being in the music industry. The answer yes. Did this person live during the 20th century? The answer is yes. Is the genre of music that the person was associated with, say hip hop or rap? No. Is the person a singer? Singer? Yes. Male or female, is the person female? Person female? No. That's ten questions, Michael. Yes, the, the clock is ticking down, but I feel like I have narrowed it down quite a bit at this point. Did the person die since you've become a professor? So, say in the last How long have you been a professor? Too forever it sounds like feels like. Let's see, recent death. And I'm going to say the answer is yes. Is the person's name Michael? The answer is yes. Alright, alright, I think I'm on to it. Is it Michael Jackson? No. Woo hoo! Of course, it's Michael Jackson. Alright, Thriller. Yes, Michael Jackson is the answer. Alright. So that was, that was very fun. And I'm very glad that I was able to solve it. [LAUGH] But it's not clear to me how, this is going to give us an algorithm for

building decision trees. Okay, so let's think about that for a second. So you asked a bunch of questions, and you asked them in a particular order. Right? So, here's a question I have for you. Why was the first question you asked me whether it was an animal or not? well, it seemed like I needed some way of narrowing down the space, and so I wanted to get as much information out of that question as I could to try to make progress towards figuring who it actually was. Right. So, animal is the first question and it was because it narrowed things down. So, your goal in asking questions was to narrow the possibilities, right? Sure, right because I only have 20 questions and then I'm, you know I'm out of it, so if I asked questions like You know, if I had said--started with, Is it someone named Michael, that would have been really bizarre. Right, and if the answer was no, it's not clear that it would tell you anything at all. So, actually, that's an interesting point. You started with animal; you could have started with Michael. And if I had said yes, that would have told you something. But if I had said No, it wouldn't have told you hardly anything at all. Right? So animal is a better attribute, or feature, to ask about than Michael as a first question. Do you agree with that? Yeah, because it could have been like a stapler or something like that and then I, the Michael question would've been pretty silly. Exactly. So what about persons? So first you asked about animal. Then you asked about person. Why person? Right, because, because again it seemed like of the space of possibilities that I could think of person would help kind of again narrow things down. That I'd if it was the answer was yes, I would be able to focus there. If the answer was no I could focus some place else. Exactly. And so I think in fact we could, we have a general statement here. That each one of these questions. Person, famous, do we know this person, personally, so on and so forth. All make sense because they further narrow down the possibilities. And that bit about further is important. Because it implies that the usefulness of a question depends upon the answers that you got in the previous questions. So even though Michael is not a particularly good first question to ask, it's a perfectly reasonable twelfth question to ask or however far down it is, given that you already know this person's, this is an animal, a person, a famous person we don't know personally who's not living, who's into music, etc., etc., etc. Okay, that make sense? Yeah. Okay. So I think then, we have the hints of an algorithm. So let's try to write down that algorithm and, and see if it matches with our intuition.

DECISION TREES: LEARNING

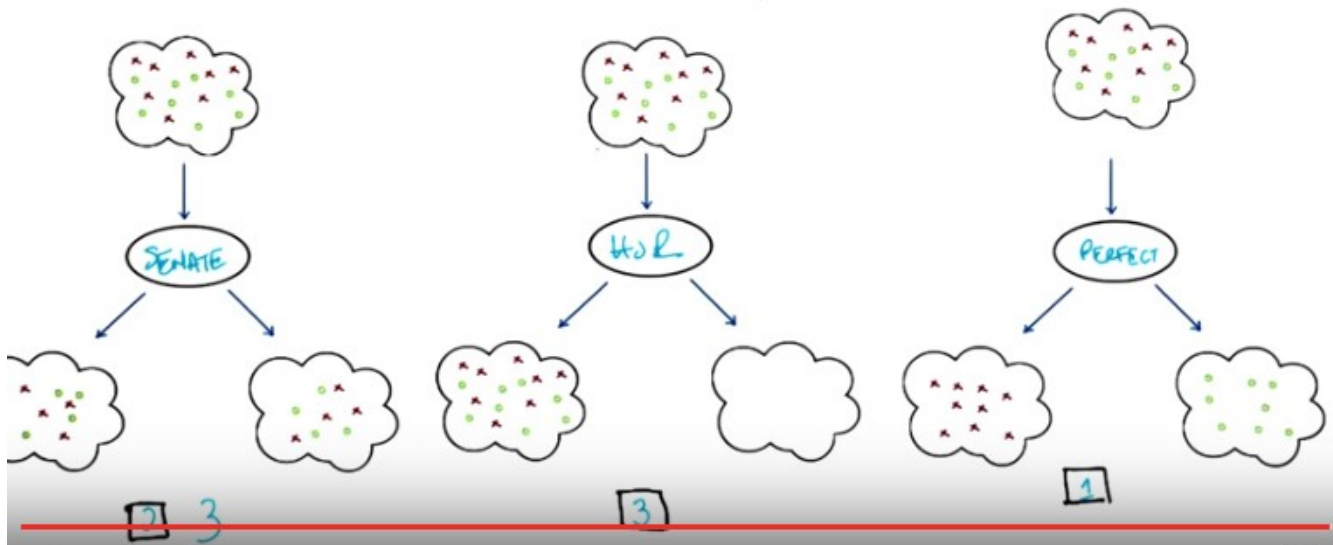
1. PICK BEST ATTRIBUTE
BEST ~ SUITS THE DATA
2. ASKED QUESTION
3. FOLLOW THE ANSWER PATH
4. GO TO 1
UNTIL GOT AN ANSWER

12. Okay, so, inspired by 20 questions let's try to write down exactly what it is that you did in going through your 20 questions to get your answer to discover Michael Jackson was the person I was

thinking about. So, what is the very first thing you did? I tried to imagine a bunch of possible answers that you could be, could have in mind. And, I tried to think of a question that would help separate them into two roughly equal chunks. Okay, so what's the way of putting that in the language that we've been using for supervised learning and classification so far? Oh I see. So if I had known in advance, here's here's 200 things that you might ask me about. And what their, what their attribute values are. What would be a question that would split that set in half, right? So, instead of just imagining a bunch of things, if I actually had a list, which I do in the case of a training set. Alright, so the first thing you did is you picked the best attribute that you could think of. And, by the way, you said something very particular here. You actually defined what best is. You said, that best is the same thing as splitting things roughly in half. So let's revisit that in a moment. Okay, so the first thing you did is you picked the best attribute. You asked a question about it and then, depending upon the answer, you went and picked another attribute right? Does that seem fair Yeah. Okay. So, we think about decision trees, a way of talking about that is that you follow the path of the answer, and then lather, rinse and repeat. [LAUGH]. You went back and pick the best attribute, asked the question, followed the answer path, so on, and so on, and you kept doing that until what? Until, I narrowed the space of possibilities to, in this case, just one item. Right, so until you got to the answer. All right. And that is an algorithm. So you pick the best attribute, and you actually define what best attribute was. You want to pick one that would somehow eliminate at least half of the things which you might worry about and keep the other half in play. You asked a specific question. You followed the path to that and then you went back and you picked another attribute and so on and so forth until you got an answer that you wanted to. That's an algorithm and that's an algorithm that we might use to actually build a decision tree. And the only difference between what you did and what you would do with learning a decision a tree is that, since you don't know in advance what the answer actually is going to be, because you don't know what specific object I might be thinking of, you have to actually follow all possible paths at each time and think of all possible best next attributes until you completely can answer any question. Does that make sense? I see. So, right, so instead of just playing the game interactively, I kind of imagine all possible ways it could go and build that whole flow chart, that whole tree. Right, so, let's see if we can do that with some pictures and I actually want to decide rather I really believe your definition of best. Okay? Sure. Alright, so, let's do that.

13. Alright, so let's take a moment to have a quiz where we really delve deeply into what it means to have a best attribute. So something Michael and I have been throwing around that term, let's see if we can define it a little bit more crisply. So, what you have in front of you are three different attributes that we might apply in our decision tree for sorting out our instances. So, at the top of the screen what you have is you have a cloud of instances. They are labelled either with red x or a green o, and that represents the label so that means that they are part of our training set, so this is what we're using to build and to learn our Decision Tree. So, in the first case you have the set of instances being sorted into two piles. There are some xs and some os on the left and some xs and some os on the right. And the second case you have that same set of data being sorted so that all of it goes to the left and none of it goes to the right. And in the third case you have that same set of data that's sorted so that a bunch of the xs end up on the left and a bunch of the os end up on the right. What I want you to do is to rank each one, where one is the best and three is the least best attribute to choose. Go.

QUIZ: "BEST" ATTRIBUTE



HOR: House of Representatives

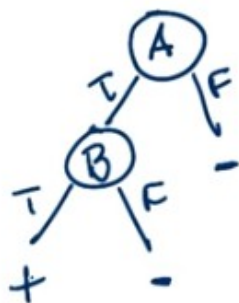
14. Okay Michael. So, are you ready to give me a ranking? Yes, yes, I think I am. Okay, well, if you give me a ranking then. So, did you say one was the best. One is the best and three is the least best. Alright, so I am really excited about the third cloud structure. The third attribute to be split on. Because, what it does. Is it takes all our x's and o's That need to have different labels and it puts them into two different buckets. One where they all get to be red x's and the other where they all get to be green o's. So I would say the far right one is ranked number one. I would agree with you and in fact I would say that infact we're done. Yeah, it's perfect. It is perfect, agreed. One is perfect. Or 3 is perfect in this case, because I gave it a one. Alright, so then, I think the worst one is also easy to pick, because if you look at the middle attribute, the attribute that's shown in the middle, we take all the data, and we put it all on the left. So we really have just kicked the can down the road a little bit. There's nothing. That this attribute splitting has done. So, I would call that the worst possible thing you could do. Which is to basically to do nothing. No, I think that's right. and, in fact, it really is doing nothing. Right. Okay. So what about the first attribute? So, I'm going to put that at, you know, if the first one is too hot and the middle one is too cold, I would say this one is, wait, no, no. [LAUGH] I was going down the Goldilocks path. So, so this one is sort of in between. In that it splits things so you have smaller sets of data to deal with, but it hasn't really improved our ability to tell the reds and the greens apart. So in fact, I'd almost want to put this as three also but I'll put it as two. Okay. I think an argument could be making it three. An argument could be made for making it two. Your point is actually pretty good, right? We have eight red things and eight green things up here. And the kind of distribution between them, sort of half red and, half red x's, half green o's, we have the same distribution after we go through this attribute here. So it does some splitting, but it's still, well you still end up with half red, half green, half x, half o. So, that's not a lot of help, but it's certainly better than doing absolutely nothing. Well is it though? I mean, it seems it could also be the case. That what we've done is that we're now splitting on that has provided no valid information, and therefore can only contribute to overfitting. That's that's a good point. That's a good point. So, do you want to change your answer to three? I don't know, what did you want the answer to be? It seems to me that the first

one and the second one are just plain bad. They are just plain bad. The question is whether one is, more bad than the other. I, I don't know. I don't know how to judge. Well I'll tell you, I would accept either two or a three as an answer here (for the first one). I think you can make an argument either way. And I think you actually made both arguments. That's very nice of you. So. Thank you. You're very welcome. So this is perfect. This is the House of Representatives and this is the Senate. [LAUGH] What do you mean they're? Oh. So, there you go. Okay. [LAUGH] Not exactly sure what you mean, but it seems somehow denigrating to our political system. It is not at all denigrating. It is, it is, I would call it incisive political satire.

DECISION TREES : EXPRESSIVENESS

BOOLEAN

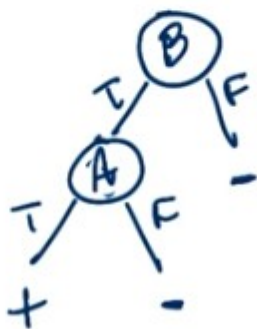
A AND B



DECISION TREES : EXPRESSIVENESS

BOOLEAN

A AND B



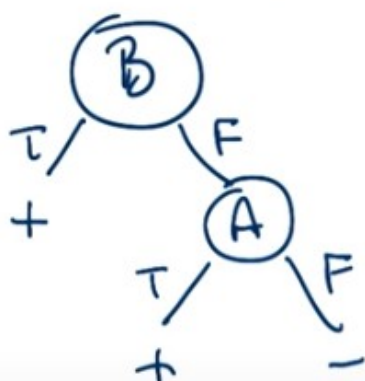
上圖的意思是: A 和 B 交換後, decision tree 還是一樣的.

15. Okay. So Michael, for the last 15 minutes or so we've been talking about decision trees, sort of in the abstract without saying too much about the kinds of functions they can actually represent. So, for the next few minutes or so I want to talk a little bit about not just decision trees in the abstract, but exactly how expressive they can be. Is that okay? Yeah, I think that would be really helpful. I think so too. So in fact, I want to look at a set of functions, and in particular I'm interested in looking at Boolean functions. Boolean. So what's your favorite simple Boolean function? Implication. What's your other favorite simple boolean function? Well I like Nor. Right. So what I just heard you say is you like And, so let's do that one. Oh, that's great. That is my favorite. All right. So, in fact, let's do very simple And. So, how does And work, right? So, you've got two attributes. Let's say A and B. And, this expression, A and B, is true when? When A and B are true. When they are both true at the same time. Right, exactly. So, how would you build a decision tree, given that there are only two attribute, A and B, to represent that function? Okay so I'd have a node that's A and B. And if that's true. Nope, you're not allowed to do that. Oh. Every node can be, have at most one attribute. All right well let's let's put A in an attribute. Okay, so here's a little node. Let's call it A. Okay. Now what? And well I mean, for it to be a node it needs to have the little two branchy things. True and false. Okay. All right, so how about true on the left and false on the right? Sure, as long as you label them. So, all right. So then A, if A is true, okay, I don't know. But oh, but if A is false, then we know that A and B must be false. Doesn't matter what B is. So we can just put a leaf under the F. That's correct. All right, I like that. Okay. What about when A is true? What, is that an F? That is an F, and that is a minus sign for false. Oh, a minus sign. I get it, okay. I thought you were just not done drawing the F yet. good. All right. So, oh yeah. On the true side then, I don't think we know. So I think we need to split on B also. Okay. So put a little B under there. All right, done. All right, and then true-false split on B. All right and so now we've got these two cases. So if B is false, then again, it doesn't matter what A was but A turns out to be true. But it's still the, the, it should be a minus sign underneath that. Okay. So it's not A and B is not true. But if A is true and B is true then A and B is true. So there should be a plus sign on the left. That's exactly right. Woo. So clearly decision trees proof by, we just drew it here, can represent the Boolean function And. Sure. [CROSSTALK]. You said something int, you said something interesting, Michael. You said it doesn't matter what A is if B is false. So what would happen if I switch, A and B around. That's the same. So if B, okay, in the beginning, we say, if B is false, it's false. If B is true, we check A. If A is false, then it's false. But if A is true, then it's true. So it actually still represents exactly the same function, A and B. Oh, because A and B is collaborative. No, commutative. Yes? No? Hello? It's one of those things. It's commutative. As opposed to associative. As opposed to what? Associative. Well I mean it, it's that too. But I mean, it's the reason that you can just switch those two things and it didn't make a difference is because they play the same role in the function. That's true in, in terms of representation of the decision trees. You know, it doesn't really matter which attribute you pick or the order in which you do them. You might get a better tree or a worse tree or a longer tree or a shorter tree. But for something simple like, two valued And, it really just doesn't matter. Okay. kind of neat, huh? Yeah.

DECISION TREES : EXPRESSIVENESS

BOOLEAN

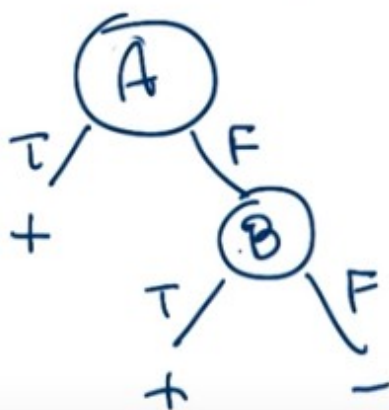
✓ A AND B
A OR B



DECISION TREES : EXPRESSIVENESS

BOOLEAN

✓ A AND B
A OR B



16. Okay, so we can do A and B. So, let's pick another function. What's your other favorite, Boolean function of two variables? Well, if we're doing two attributes, you know, or is the other really nice one. I like or. So, let's see. Do you think, now that we've shown that we can do A and B, could we do A or B? I feel like we could do A or B. because there's that nice De Morgen's Law relationship between them but but let's just, how about this, can you erase the tags at the bottom? Like that? Oh, yeah but that's not going to work, is it? So, so if B and A are both true, so the left branch, then the, the, the leaf should be plus? Yes. And if B is true and A is false then the tag should be plus. Yes. And if

B is false, I want to say false but we don't know because A could be true and A or B is true if either of the two of them are true. Oh okay, so maybe the mistake here is just trying to use exactly the same thing. So let's just erase it. All right. And start from scratch. So what would you do starting from scratch? I'd split on B again. Okay. True False or False True. True False. True False. All right, and now if B is true. We know that A or B is true, so we can put a plus under the left side. Right. But if B is false. Mm-hm. Then we need to split on A. Okay. And if A is true, then we're golden, we get the Or is true. And if A is false, then it is false. Very good, and that represents A or B. That represents the function, the logical function A or B, yeah. Right. What happens if I swap A and B around, does it still work? I mean, my, my, since they're collaborative or commutative. Since they're commutative I want to say it shouldn't make a difference, but let's just double check that. So if A is true then the output's true. If A is false and B is true then the output's true. And if A and B are both false, then the answer's false, yeah, totally. Excellent. And, you know, if I hadn't erased it you would see that the two trees actually look very, very much alike. They're sort of mirror each, of each other. If you just swapped around. Yeah the, yeah exactly, they're mirrors of one another.

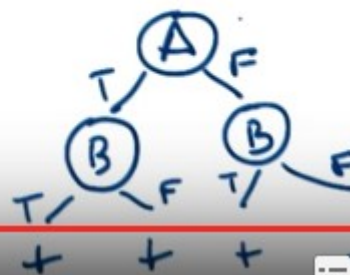
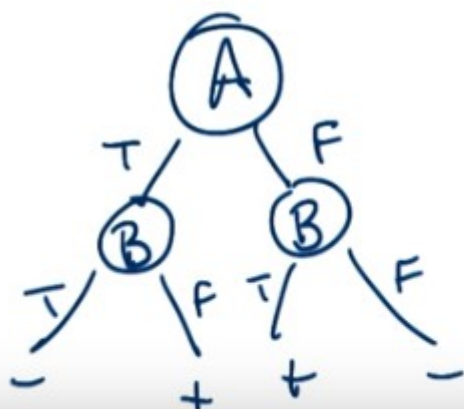
DECISION TREES : EXPRESSIVENESS

BOOLEAN

✓ A AND B

✓ A OR B

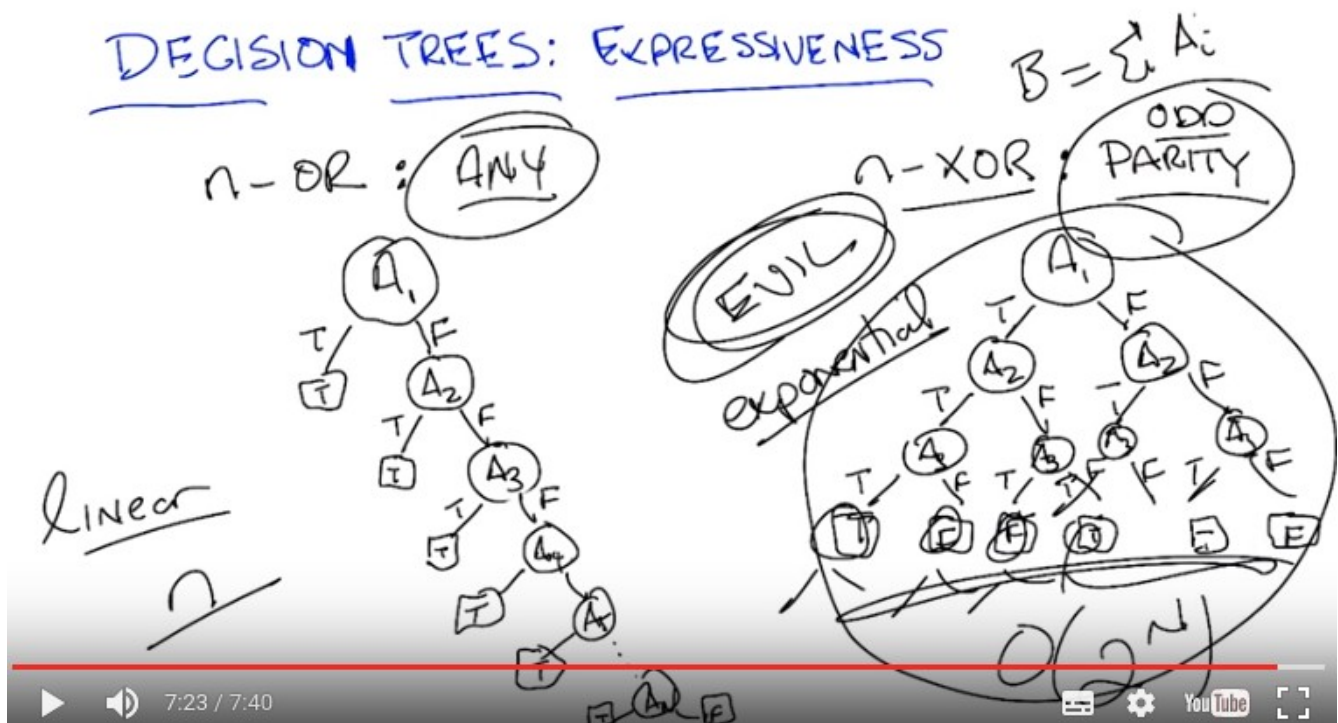
✓ A XOR B



上圖左邊的 decision tree 表示的 XOR (A XOR B 等價於 $A \neq B$), 右邊的 decision tree 表示的 OR.

17. [LAUGH] Okay, let's pick one more Boolean function to do. What function am I thinking of? XOR is always good. Let's do XOR. So what does XOR mean? Remind me. Okay. XOR is exclusive or, which means it's true if A is true or B is true, but not if they're both true, so it has elements of both or and and in it. Right. And I think of XOR usually when I'm, when I'm, like, playing with light switches in my house. If I have a, a light that's activated by two different light switches, it's usually an XOR function. If one is up, the light's on, if the other one's up, the lights on, but if they're both up, the light goes back off. Right. The other thing when I think of XOR is when people say or most of the time when people say or, when they're talking, they mean xor. Really? Yeah. [So a lot of times when](#)

people say or in English, they really mean xor. They say, well do you want to go to the movies or do you want to go swimming? Do want to eat chicken or do you want to eat fish? And really, they're saying either or. I see, you want one or the other, but you can't have chicken and fish or go swimming and the movies. No, those things are not possible to do together. Got it. Okay. All right, so how would you do XOR? We got, we still have our- Well, I would start with the- Or, because it's a lot like or. So, what would you want to do? Okay, so to do XOR, we can split on A. Okay. And there's a true branch and a false branch. And what happened with and and or at this point is, there is at least one branch where we actually knew the answer, at this point, but I don't think that's true here. That's right. So, so if A is true, the output might be true or false. It depends on B. And if A is false, the output might be False or True. It depends on B. So I- This is exactly right. So I guess in both cases we still have to split on B. Okay. All right. So, now it should be relatively easy in the sense that there's a separate leaf for all possible inputs. And so we can just write the XOR function on the bottom. So if A and B are both true, then the output is false because it's exclusive. If A is true and B is false, then it should be true, because A is true. If A is false and B is true, then it should be true because B is true. And if A and B are both false then it should be false. That's right. And by the way, if you were to think about it for a little while, it would probably occur to you that's [this tree \(左邊的 decision tree\) is just a another representation of the full truth table](#). Very nice. And in fact, if we wanted to do or again, we could say, well I was very smart last time, but I could actually write or like this. And then just fill out the values at the bottom. If A is true and B is true, then yes. If A is true and B is false, then yes. If A is false and B is true, then yes. If A is false and B is false, then no. [And this\(右邊的 decision tree\) is a perfectly legitimate a decision tree to represent or because it basically is just another simple representation of the truth table](#), but as we pointed out when did last time, it's more of a decision tree than we actually need. I can see that. Because in particular, we don't actually need this. All right. So this little difference here between writing out the entire truth table on the left, as we did for XOR, and not having to write out the entire decision tree on the right for or, actually isn't going to turn out to matter when we try to do things either more complicated or with more attributes that we did for the simple and, or, and XOR. Would you like to see? Yeah, that sounds really cool. Beautiful.



18. So, we saw before when we looked at AND and OR versus XOR that in the case of AND and OR we only needed two nodes but in the case of XOR we needed three. The difference between two and three is not that big, but it actually does turn out to be big if you start thinking about having more than simply two attributes. So, let's look at generalized versions of OR and generalized versions of XOR and see if we can see how the size of the decision tree grows differently. [So in the case of an \$n\$ version of OR. That is we have \$n\$ attributes as opposed to just two. We might call that the any function.](#) That is a function where any of the variables are true then the output is true. We can see that the decision tree for that has a very particular and kind of interesting form. Any ideas Michael about what that decision tree looks like? So, well. So going off of the way you described OR in the two case. We can start with that. And you. You pick one of the variables. And if it's true then yeah. Any of them is true since the leaf is true. What happens if it's false? Well, then we have to check what everything that's left. [LAUGH] So then we move on to one of the other attributes like A2. mm hm. And again, if it's true, it's true and if it's false then we don't know. Look at A3. Good idea. This could take some time. Yes, oh that was actually an interesting point. Let's say if there were only three, we would be done. Right? Right. But wait, what if there were five? Then we need one more node. What if there were n ? Then we need n minus 4 more nodes. Right, so what you end up with in this case is a nice little structure around the decision tree. And [how many nodes do we need?](#) Looks like one for each attribute, so that would be n . [\$n\$ nodes](#), exactly right. [So we have a term for this sort of thing, the size of the decision tree is, in fact, linear.](#) In n . And that's for any. [Now what about an \$n\$ version of XOR?](#) Mm. So XOR is, if one is true but the other one's not true then it's true. And if they're both true. Yeah I don't. It's not clear headed. Generalize that. So why not hmm. So, while [the usual general version of this we like to think of as parity. All parity is a way of counting, so there's usual two forms of parity that we worry about. Either even parity or odd parity. So let's pick one, it doesn't matter. Let's say. Odd. I like that, we'll do odd parity.](#) And all that works out to be in this case is, [if the number of attributes that are true is an odd number, then the output of the function is true, otherwise it's false \(這其實是對 \$n\$ -XOR 的定義\).](#) Got it? Got it. Okay, so, how would we make that decision tree work? Ooh. Well, we got to split on something and there all the same, so let's split on A1 again. Okay. So what do we do if A1 is true, versus being false. We don't know much if A1 is true. We have to look at everybody else. Right. So let's look at A2. What if A2 is true versus false? Well if A1 and A2 are true then, then the output is going to be whatever the parity of all the remaining variables are. So you still have to do that. Uh-huh, yup. And I'm already running out of room, so let's pretend there's only three variables. What's the output? [LAUGH] All right, so the far left. Is there's three trues which is odd so the output is true. Yep. The next leaf over, only two trues. A1 is true, A2 is true, but A3 is false, so that's two trues which is even so the answer's false. Um-huh. Is this going to, is this pattern continuing? Now we've got. No, so then it's false again because we've got two trues and a false to get to the next leaf. Mm-hm. And we've got one true to get to the next leaf so that's true. Oh, that looks like XOR. It looks just like XOR. In fact, each one of these sub trees is kind of a version of XOR isn't it? Now what we have is, we have to do the same thing on the right. So we gotta redo A2, and we're going to be in the same situation before. And we're going to start drawing on top of each other because. [LAUGH] there's just not enough room. Hm, so, what's the answer to the one on the very right. Where all of them is false. All of them are false. So that's, an even number of trues. Zero is even. So that's false. Okay, so in the case where only A3 is true, it's true and we just keep going on and on and on again. Now imagine what would happen, in fact let me ask you Michael, what would happen if we had four attributes instead of three. Then we would be really tired of this game. Yes, and I am already tired of this game so the question is, can you. We get a whole another, a whole other level of this tree. Yep. We have the, it just goes on and on and on and nobody wants to think about it anymore. [LAUGH] So, how many nodes do you think there are? Well, for three there was one, two, three, four, five, six, seven. Which seems suspiciously like one less than the power of two. Mm-hm. And that is exactly right. You need more or

less 2 to the n nodes. Or. 2 to the n , maybe, minus 1. Yeah. So let's just say big $O(2^n)$. Everyone watching this is a computer scientist so they know what they're doing. Okay so, you need an exponential therefore, as opposed to linear number of nodes. Gad. Yeah, so you very, very quickly run out of room here. You very, very quickly have a really, really big tree because it's growing exponentially. So, XOR is an exponential problem and is also known as hard. Whereas OR, at least in terms of space that you need, it's a relatively easy one. This is linear. We have another name for exponential and that is evil. Evil, evil, evil. And it's evil because it's a very difficult problem. There is no clever way to pick the right attributes in order to give you an answer. You have to look at every single thing. That's what makes this kind of problem difficult. So, just as a general point, Michael, I want to make, is that we hope that in our machine learning problems we're looking at things that are more like any than we are looking at things that are more like parity. Because otherwise, we're going to need to ask a lot of questions in order to answer the, the parity questions. And we can't be particularly clever about how we do it. Though, if we were kind of clever and added another attribute, which is like, the sum of all the other attribute values (MO: 即重新用一個量 $B = \sum A_i$, 這樣的話, decision tree 就只有 B 一個 node 了, 因為可以直接從 B 的奇偶得出結果), that would make it not so bad again. So maybe it's just a, it's just a kind of, bad way of writing the problem down. Well, you know, what they say about AI is that the hardest problem is coming up with a good representation (意思是: 如何找 good representation 才是最難的). So what you just did is, you came up with a better representation, where you created some new pair, new variable. Let's call it B , which is just the sum of all of the A s, where we pretend that I don't know, true is one and false is zero. This is actually a really good idea. It's also called cheating. Because you got to solve the problem by picking the best representation in the first place. But you know what? It's a good point, that in order for a machine running to work, you either need an easy problem or you need to find a clever way of cheating. So, let's come back and think about that throughout all the rest of the lessons. What's the best way to cheat?

DECISION TREES: EXPRESSIVENESS

→ XOR is hard

→ n attributes (boolean)

→ How many trees?

→ output is boolean

$O(2^n!)$ nodes

answers

A LOT

上圖中的 $O(n!)$ 不重要

19. All right, so what that last little exercise showed is that XOR, in XOR parody, is hard. It's exponential. But that kind of provides us a little bit of a hint, right? We know that XOR is hard and we know that OR is easy. At least in terms of the number of nodes you need, right? But, we don't know, going in, what a particular function is. So we never know whether the decision tree that we're going to have to build is going to be an easy one. That is something linear, say. Or a hard one, something that's exponential. So this brings me to a key question that I want to ask, which is, exactly how expressive is a decision tree. And this is what I really mean by this. Not just what kind of functions it kind of represent. But, if we're going to be searching over all possible decision trees in order to find the right one, how many decision trees do we have to worry about to look at? So, let's go back and look at, take the XOR case again and just speak more generally. Let's imagine that we once again, we have n attributes. Here's my question to you, Michael. How many decision trees are there? And look, I'm going to make it easy for you, Michael. They're not just attributes, they're Boolean attributes. Thanks. Okay? And they're not just Boolean attributes, but the output is also Boolean. Got it? Sure. But how many trees? So it's, I'm going to go with a lot. Okay. A lot. Define a lot. Define a lot. So, alright, well, **there's n choices for which node to split on first. Yeah. And then, for each of those, there's $n-1$ to split on next. So I feel like that could be an n factorial kind of thing.** Maybe. I like that. And then, even after we've done all that, then we have an exponential number of leaves. And for each of those leaves, we could fill in either true or false. So it's going to be exponential in that too. Hm, so let me see if I got that right. So you said we have to pick each attribute at every level. And so you see something that you think is probably going to be, you know? Some kind of commutatorial question here. So, let's say n factorial, and that's going to just build the nodes. That's just the nodes. Well, once you have the nodes, you still have to figure out the answers. And so, this is exponential because factorial is exponential. And this is also exponential. Huh. So let's see if we can write that down.

DECISION TREES: EXPRESSIVENESS

- XOR is hard
- n attributes (boolean)
- How many trees?
- output is boolean

TRUTH TABLE

A_1	A_2	A_3	...	A_n	output
T	T	T	...	T	
T	T	T	...	F	
F	T	T	...	T	
...					

A LOT

How MANY ROWS?

2^n

So let me propose a way to think about this, Michael. You're exactly right the way you're thinking. So, let's see if we can be a little bit more concrete about it. So, we have Boolean inputs and we have Boolean outputs, so this is just like AND, it's just like OR, it's just like XOR, so, whenever we're dealing with Boolean functions, we can write down a truth table. So let's think about what the truth table looks like in this case. [SOUND] Alright, so, let's look at the truth table. So what a truth table will give me is, for, the way a truth table normally works is you write out, each of the attributes. So, attribute one, attribute two, attribute three, and dot dot dot. And there's n of those, okay? We did this a little earlier. When we did our decision tree. When we tried to figure out whether I was on a hot date or not. And then you have some kind of output or answer. So, each of these attributes could take on true or false. So one kind of input that we may get would be say all trues. Right? But we also might get all trues, except for one false at the end. Or maybe the first one's false and all the rest of them are true, and so on, and so forth. And each one of those possibilities is another row in our table. And that can just go on for we don't know how long. So we have any number of rows here and my question to you is how many rows? Go. (圖在第 21 段之後)

20. All right, so what that last little exercise showed is that XOR, in XOR parody, is hard. It's exponential. But that kind of provides us a little bit of a hint, right? We know that XOR is hard and we know that OR is easy. At least in terms of the number of nodes you need, right? But, we don't know, going in, what a particular function is. So we never know whether the decision tree that we're going to have to build is going to be an easy one. That is something linear, say. Or a hard one, something that's exponential. So this brings me to a key question that I want to ask, which is, exactly how expressive is a decision tree. And this is what I really mean by this. Not just what kind of functions it kind of represent. But, if we're going to be searching over all possible decision trees in order to find the right one, how many decision trees do we have to worry about to look at? So, let's go back and look at, take the XOR case again and just speak more generally. Let's imagine that we once again, we have n attributes. Here's my question to you, Michael. How many decision trees are there? And look, I'm going to make it easy for you, Michael. They're not just attributes, they're Boolean attributes. Thanks. Okay? And they're not just Boolean attributes, but the output is also Boolean. Got it? Sure. But how many trees? So it's, I'm going to go with a lot. Okay. A lot. Define a lot. Define a lot. So, alright, well, there's n choices for which node to split on first. Yeah. And then, for each of those, there's n minus 1 to split on next. So I feel like that could be an n factorial kind of thing. Maybe. I like that. And then, even after we've done all that, then we have an exponential number of leaves. And for each of those leaves, we could fill in either true or false. So it's going to be exponential in that too. Hm, so let me see if I got that right. So you said we have to pick each attribute at every level. And so you see something that you think is probably going to be, you know? Some kind of commutatorial question here. So, let's say n factorial, and that's going to just build the nodes. That's just the nodes. Well, once you have the nodes, you still have to figure out the answers. And so, this is exponential because factorial is exponential. And this is also exponential. Huh. So let's see if we can write that down. So let me propose a way to think about this, Michael. You're exactly right the way you're thinking. So, let's see if we can be a little bit more concrete about it. So, we have Boolean inputs and we have Boolean outputs, so this is just like AND, it's just like OR, it's just like XOR, so, whenever we're dealing with Boolean functions, we can write down a truth table. So let's think about what the truth table looks like in this case. [SOUND] Alright, so, let's look at the truth table. So what a truth table will give me is, for, the way a truth table normally works is you write out, each of the attributes. So, attribute one, attribute two, attribute three, and dot dot dot. And there's n of those, okay? We did this a little earlier. When we did our decision tree. When we tried to figure out whether I was on a hot date or not. And then you have some kind of output or answer. So, each of these attributes could take on true or false. So one kind of input that we may get would be say all trues. Right? But we also might get all trues, except for one

false at the end. Or maybe the first one's false and all the rest of them are true, and so on, and so forth. And each one of those possibilities is another row in our table. And that can just go on for we don't know how long. So we have any number of rows here and my question to you is how many rows? Go.

21. Alright, so here's the question for you. We know we have 2DN, different possible instances we might see. That is two to the end, different ways we might assign different values to the attributes. But, that still doesn't tell us how many decision trees we may have, or how many different functions we might have. So, if we have 2DN rows, here's my question. How many different ways might we fill out. This column over here of outputs? Remember, an output can be either true or false. Go.

DECISION TREES: EXPRESSIVENESS

- XOR is hard
- n attributes (boolean)
- How many trees?
- output is boolean

A LOT $2^{(2^n)}$

TRUTH TABLE

A_1	A_2	A_3	...	A_n	output
T	T	T	...	T	
T	T	T	...	F	
F	T	T	...	T	
			...		
			...		
			...		

2^n rows → How MANY WAYS TO FILL IN THE OUTPUTS?
 $2^{2^n} \neq 4^n$

DECISION TREES: EXPRESSIVENESS

→ XOR is hard

→ n attributes (boolean)

→ How many trees?

→ output is boolean

$n=6, 2^{2^n}$

18 466 744 073 709 551 616

TRUTH TABLE

A_1	A_2	A_3	...	A_n	output
T	T	T	...	T	
T	T	T	...	F	
F	T	T	...	T	
			...		
			...		
			...		
			...		

2^n rows →
How MANY WAYS TO
FILL IN THE OUTPUTS?
 $2^{2^n} \neq 4^n$

22. Okay, Michael, what's your answer? Alright. So. Again, a lot feels like a good answer, it's already written down on the left. But it's also wait, wait, maybe we can quantify this. So if it were. Maybe one way to think about this is if each of the, each of those empty boxes there, is either true or false. It's kind of like a bit. And we're asking how many different bit patterns can we make? And in general, it's two to the number of positions, but here the number of positions is 2 to the n . So it ought to be 2, to the 2 to the n . Which is that the same as 4 to the n ? No. Okay. But you're right. It's 2 to the 2 to the n . So it's a double exponential and it's not the same thing as 4 to the n th. It's actually 2 to the 2 to the n th. Now how big of a number do you think that is Michael? I'm going to go again to my go to place and just say a lot. It is, in fact, a lot and I'm going, I actually, I'm going to look over here, and I'm going to tell you. That for even a small value of n , this gets to be a really big number. So for, for one, it's 2 to the 2 to the 1, which is 4. That's not a big number. That's true. What about two? For two, it's 2 to the 2 to the 2. So 2 to the 2 is 4, so it's 2 to the 4, which is 16. What about three? Alright, so that's two to the 8th, which is 256? Mm-hm. So that's growing pretty fast, don't you think? Sure, but those aren't big numbers yet. What if I told you, [LAUGH] that for n equals 6, 2 to the 2 to the n was, I'm going to start writing it, okay? 18466744073709551616. Holy monkeys. Yes, that is in fact the technical term for this number, it's a holy monkey. It is a very, very big number. So 2 to the n grows very fast. We already called that evil. 2 to the 2 to the n is a double exponential and it's super evil. It grows very, very, very, very, very fast. So what's the point of this exercise, Michael? It's, it's to point that the space of decision trees, the hypothesis space that we've chosen, is very expressive because there's lots of different functions that you can represent. But that also means we have to have some clever way to search among them. And that gets us back to our notion of an algorithm with actually going to very smartly go through and pick out which decision tree. Because if we aren't very smart about it and we start eliminating whole decision trees along the way. Then we're going to have to look it to billions upon, billions upon, billions upon, billion upon, billion of possible decision choice.

103

Loop:

- $A \leftarrow$ best attribute
- Assign A as decision attribute for node
- For EACH VALUE OF A create a descendant of node
- SORT TRAINING EXAMPLES TO LEAVES
- IF EXAMPLES PERFECTLY CLASSIFIED STOP
- ELSE ITERATE OVER LEAVES

$$\text{GAIN}(S, A) =$$

$$\text{Entropy}(S) -$$

$$\sum_v \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$- \sum_v P(v) \log P(v)$$

上圖底部的 IF 一句是 If examples perfectly classified
 右下角那個藍色公式是 entropy 的公式

From online:

Information gain :

- The entropy typically changes when we use a node in a decision tree to partition the training instances into smaller subsets
- Information gain is a measure of this change in entropy
- Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with $A = v$, and $\text{Values}(A)$ is the set of all possible values of A, then

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$
 (recall $|S|$ denotes the size of set S)

對於 $\text{Entropy}(S)$, 我的理解是: 在用 entropy 的公式算 $\text{Entropy}(S)$ 時, $P(v)$ 即這樣一個狀態的概率: 這個狀態即: S 中所有 attribute 都各自取一些值, 這些值的集合即 v.

23. So now, we have an intuition of best, and how we want to split. We've, we've looked over, Michael's proposed, the high-level algorithm for how we would build a decision tree. And I think we have enough information now that we can actually do, a real specific algorithm. So, let's write that down. And the particular algorithm that Michael proposed is a kind of generic version of something that's called ID3. So let me write down what that algorithm is, and we can talk about it. Okay, so here's the ID3 algorithm. You're simply going to keep looping forever until you've solved the problem. At each step, you're going to pick the best attribute, and we're going to define what we mean by best. There are a couple of different ways we might, we might define best in a moment. And then, given the

best attribute that splits the data the way that we want, it does all those things that we talked about, assign that as a decision attribute for node. And then for each value that the attribute A can take on, create a descendent of node. Sort the training examples to those leaves based upon exactly what values they take on, and if you've perfectly classified your training set, then you stop. Otherwise, you iterate over each of those leaves, picking the best attribute in turn for the training examples that were sorted into that leaf, and you keep doing that. Building up the tree until you're done. So that's the ID3 algorithm. And the key bit that we have to expand upon in this case, is exactly what it means to have a best attribute. All right so, what is exactly, what exactly is it that we mean by best attribute? So, there are lots of possibilities, that you can come up with. The one that is most common, and the one I want you to think about the most, is what's called information gain. So information gain is simply a mathematical way to capture the amount of information that I want to gain by picking particular attribute, funnily enough. But what it really talks about is the reduction in the randomness, over the labels that you have with set of data, based upon the knowing the value of particular attribute. So the formula's simply this. 「The information gain over S and A where S is the collection of training examples that you're looking at, and A, as a particular attribute」, is simply defined as the entropy, with respect to the labels, of the set of training examples, you have S, minus, sort of, the expected or average entropy that you would have over each set of examples that you have with a particular value. Does that make sense to you, Michael? So what we're doing, we're picking an attribute and that attribute could have a bunch of different values, like true or false, or short, medium, tall? Right. And. And that's represented by v. Okay, each of those is a different v. And then we're saying okay, for over those leaves, we're going to do this entropy thing again. Mm-hm. And we right. So what, and what is entropy? entropy. So, we'll talk about entropy later on in the class in some detail and define it exactly and mathematically. And some of you probably already know what, what entropy is, but for those of you who don't, it's exactly a measure of randomness. So if I have a coin, let's say a two-headed coin. It can be heads or tails, and I don't know anything about the coin except that it's probably fair. If I were to flip the coin, what's the probability that it would end up heads versus tails? A half. It's a half, exactly, if it's a fair coin it's a half. Which means that I have no basis, going into flipping the coin, to guess either way whether it's heads or it's tails. And so that has a lot of entropy. In fact it has exactly what's called one bit of entropy. On the other hand, let's imagine that I have a coin that has heads on both sides. Then, before I even flip the coin, I already know what the outcome's going to be. It's going to come up heads. So what's the probability of it coming up with heads? It's. One. One. So that actually has no information, no randomness, no entropy whatsoever. And has zero bits of entropy. So, when I look at this set of examples that I have, and the set of labels I have, I can count the number that are coming up, let's say, red x's. Versus the ones that are coming up green o's. And if those are evenly split, then the entropy of them is maximal, because if I were to close my eyes and reach for an instance, I have no way of knowing beforehand whether I'm more likely to get an x or I'm more likely to get an o. On the other hand, if I have all the x's in together, then I already know before I even reach in that I'm going to end up with an x. So as I have more of one label than the other the amount of entropy goes down. That is, I have more information going in. Does that make sense, Michael? I think so. So, is there, can, maybe we can say what the formula is for this, or, or? Sure. What is the formula for it? You should remember. It's, if we have, well, I'm not sure what the notation ought to be with these S's but it has something to do with $\log P$ log, no wait, it's $P(\log)P$. Mm-hm. So the actual formula for entropy, using the same notation that we're using for information game is simply the sum, over all the possible values that you might see, of the probability of you seeing that value, times the log of the probability of you seeing that value, times minus one. And I don't want to get into the details here. We're going to go into a lot more details about this later when we get further on in the class with randomize optimization, where entropy's going to matter a lot. But for now, I just, you have, I want you to have the intuition that this is a measure of information. This is the measure of randomness in some variable that you haven't seen. It's the likelihood of you already knowing what you're going to get if you close your eyes and

pick one of the training examples, versus you not knowing what you're going to get. If you close your eyes and you picked one of the training examples. Okay? Alright. So, well, so, okay, so then in the practice, trees that you had given us before, it was the case that we worked, we wanted to prefer splits that I guess, made things less random, right? So if things were all mixed together, the reds and the greens, after the split if it was all reds on one side and all greens on the other. Then each of those two sides would have very, what? They would have very low entropy, even though when we started out before the split we had high entropy. Right, that's exactly right. So if you, if you remember the, the, three examples before. One of them, it was the case that all of the samples (x and o) went down the left side of the tree. So the amount of entropy that we had, didn't change at all. So there was no gain in using that attribute. In another case, we split the data in half. But in each case, we had half of the x's and half of the o's together, on both sides of the split. Which means that the total amount of entropy actually didn't change at all. Even though we split the data. And in the final case, the best one, we still split the data in half, but since all of the x's ended up on one side and all of the o's ended up on the other side, we had to entropy or no randomness left whatsoever. And that gave us the maximum amount of information gain (why?). So is that how we're choosing the best attribute? The one with the maximum gain? Exactly. So the goal is to maximize over the entropy gain. And that's the best attribute.

103 : BIAS

RESTRICTION BIAS : H
PREFERENCE BIAS : $n \in H$

INDUCTIVE BIAS 

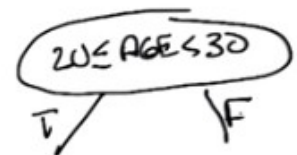
- o GOOD SPLITS AT TOP
- o CORRECT OVER INCORRECT
- o SHORTER TREES

24. So, we've got a whole bunch of trees we have to look at, Michael. And were going to have to come up with some clever way to look through them. And this gets us back, something that we've talked about before, which is the notion of bias. And in particular, the notion of inductive bias. Now, just as a quick refresher, I'm want to remind you that there is two kind of biases we worrying about when we think about algorithms that are searching through space. One is what's called a restriction bias. The other is called preference bias. So a restriction bias is nothing more than the hypothesis set that you actually care about. So in this case, with the decision trees, the hypothesis set is all possible decision trees. Okay? That means we're not considering, $y = 2x + 3$. We're not considering quadratic equations. We're not considering non-boolean functions of a certain type. We're only considering decision trees, and all that they can represent. And nothing else. Okay? So that's already a restriction bias and it's important. Because, instead of looking at the infinite number uncountably infinite number of functions that are out there, that we might consider. We're only going to consider those that can be represented by a decision tree over in, you know, all the cases we've given so far discrete variable. But a preference bias is something that's just as important. And it tells us what source of hypotheses from this hypothesis

set we prefer, and that is really at the heart of inductive bias. So Michael, given that, what would you say is the inductive bias of the ID3 algorithm? That is, given a whole bunch of decision trees, which decision trees would ID3 prefer, over others? So, it definitely tries, since it's, since it's making its decisions top down. It's going to be more likely to produce a tree that has basically good splits near the top than a tree that has bad splits at the top. Even if the two trees can represent the same function. Good point. So good splits near the top. Alright. And you said something very important there Michael. Given two decision trees that are both correct. They both represent the, the function that we might care about. It would prefer the one that had the better split near the top. Okay, so any other preferences? Any other inductive bias on the, on the ID3 algorithm. It prefers ones that model the data better to ones that model the data worse. Right. So this is one that people often forget: it prefers correct ones to incorrect ones. So, given a tree that has very good splits at the top but produces the wrong answer. It will not take that one over one that doesn't have as good splits at the top, but does give you the correct answer. So that's really, those are really the two main things that are the inductive bias for ID3. Although, when you put those two together, in particular when you look at the first one, there's sort of a third one that comes out as well, which is ID3 algorithm tends to prefer shorter trees to longer trees. Now, that preference for shorter trees actually comes naturally from the fact that you're doing good splits at the top. Because you're going to take trees that actually separate the data well by labels, you're going to tend to come to the answer faster than you would if you didn't do that. So, if you go back to the example where we went before, where one of the attributes doesn't split the data at all, that is not something that ID3 would go for, and it would in fact create a longer and unnecessarily longer tree. So it tends to prefer shorter trees over longer trees. So long as they're correct and they give you good splits near the top of the tree.

DECISION TREES: OTHER CONSIDERATIONS

- CONTINUOUS ATTRIBUTES?
e.g. AGE, WEIGHT, DISTANCE



25. Alright. So, we've actually done pretty well. So through all of this, we finally figured out what decision trees actually are. We know what they represent. We know how expressive they are. We have an algorithm that lets us build the decision trees in an effective way. We've done just about everything there is to do with decision trees, but there is still a couple of open questions that I want to think about. So, here's a couple of them and I want you to, to think about and then we'll discuss them. So, so far all of our examples that we've used. All the the things we've been thinking about for good pedagogical reasons. We had not only discrete outputs but we also had discrete inputs. So one question we might ask ourselves, is what happens if we have, continuous attributes? So Michael, let me ask you this. Let's say we had some continuous attributes. We weren't just asking whether someone's an animal or whether they're human or whether it's raining outside or we really cared about age or weight or distance or anything else that might have a continuous attribute. How are we going to make that work in a decision

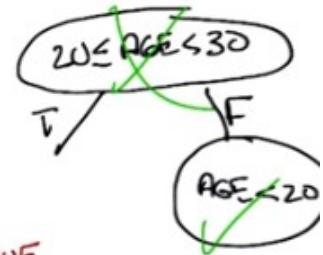
tree? Well, I guess the literal way to do it would be for something like age to have a branching factor that's equal to the number of possible ages. Okay, so that's one, one possibility. So we stick in age and then we have one. 1.0, we have one for 1.1, we have one for 1.11, we have one for 1.111. Ahh, I see. Alright. Well, at the very least, okay. Heheheh. What if, what if we only included ages that were in the training set? Presumably there's at least a finite number of those. Oh, we could do that. We could just do that, except what are we going to do then when we come up with something in the future that wasn't in the training session. Oh, right. Can we look at the testing set? No were not allowed to look at the testing set. That is cheating, and not the kind of good cheating that we do when we pick good representation. Okay, fair enough. Well we could, we could Ranges. What about ranges? Isn't that the way we cover more than just individual values? Give me an example. Say ages you know, in the 20s. Okay, so, huh. How would we represent that with a decision tree? Let's say in the 20s. Age. How we do that. You could do like age, element sign, bracket. 20 comma 21, or, or 29 or 30 right per end. Yeah it's too much. Why don't I just say age Is between less is, let's see, greater than or equal to, 20 and, less than 30. And just draw a big oval for that. Alright? So that's a range, so that's all numbers between, 20 and 30 inclusive of 20 but not 30 Right. Yeah. And what's good about that is that's a question. You are either in your 20s or you are not. So, the output of that is actually true or false. So, I guess the good news there is that now we know how to evaluate attributes like that because we have a formula from [UNKNOWN] three that tells you what to do. But seems like there's an awful lot of different ones to check. Right, and in fact if it's truly a continuous variable, there are in principal an infinite number of them checked. But we can do now the sort of cheating you wanted to do before. We can just look at the training set, and we could try to pick questions that cover the sorts of data in the training set. So, for example, if all of the values are in the 20s, then there is no point of even asking the question. You will start just [UNKNOWN] instead upon values that were, say less than 25 or greater than 25, and you could imagine all kinds of ways where you might do that. You might look at all of the values that show up in the training set, and say well, I am going to do a binary search. So, I am just going to create an attribute for Less than half of whatever is in the training set or greater than half of whatever the range is in the training set. Does that make sense? Yeah, that's clever. Right. Thank you. I just made that up on the spot. Okay, so you do those sorts of things and that's how you would deal with continuous attributes. That brings me to a next question, I'm going to actually do this as a quiz because I want an answer from our audience.

DECISION TREES: OTHER CONSIDERATIONS

- CONTINUOUS ATTRIBUTES?
e.g. AGE, WEIGHT, DISTANCE



- DOES IT MAKE SENSE TO REPEAT AN ATTRIBUTE ALONG A PATH IN THE TREE?



☐ TRUE
☐ FALSE

Answer: True

26. So, here's the next question I want to ask you, simple true or false question. Does it make sense to repeat an attribute along any given path in the tree? So, if you we pick some attribute like A, should we ever ask a question about A again? Now, I mean something very specific about, by that. **I mean, down a particular path of the tree, not just anywhere else in the tree.** So, in particular, I mean this. So, I ask a question about A, then I ask a question about B, and then I ask a question about A again. That's the question I'm asking. Not whether A might appear more than once in the tree. **So, for example, you might have been the case where A shows up more than once in the tree, but not along the same path.** So, in the second case over here, A shows up more than once, but they really don't really have anything to do with one another because once you've answered B, you will only ever ask the question about A once. So, my question to you is, does it make sense to repeat A more than once along a particular path in the tree? Yes or no? Answer the question.

27. Okay, Michael, what's the answer? So, alright. Does it make sense to repeat, an attribute along a path in the tree? So, it seems like it could be no, [SOUND] in that, you know, if we're looking at attributes like, you know, is a true, then later we would ask again is a true because we would already have known the answer to that. Right, and by the way, information gain will pick that for you automatically. It doesn't have to be a special thing in the algorithm, if, if, you consider, an attribute that you've already split on, then you're not going to gain any information, so it's going to be the worst thing, to split on. Exactly. Alright, but it, Okay, doing good. But it seems like maybe you're trying to lead us on because, this we're in the continuous attributes portion of our show. Okay, well what's the answer there? Is the answer not also false? **Well we wouldn't want to ask the same question, about the same attribute. So, we wouldn't have age, between 20 and 30, and then later ask again, age, between 20 and 30. But, maybe we want to ask, you know, given that we are less than 20, we're, are we teenagers or not, so we might have a different range, on age later in the tree.** So, that's exactly right, Michael. So, the answer is no, **it does not make sense, to repeat an attribute along a path of the tree, for**

discrete, value trees. However, for continuous [UNKNOWN] attributes, it does make sense. Because, what you're actually doing, is asking a different question. So, one way to think about this, is that the question is age in the 20's or not. Is actually, a discrete valued attribute that you've just created, for the purposes of the decision tree. So, asking that question doesn't make sense but asking a different question, about age, does in fact make sense. So, once you know, that you are not in the 20's you might ask well am I less than, 20 years old? Maybe a teenager or am I greater than 40. How old am I, 44? Greater than 44, in which case, I'm old. So if it's, if you ask, [LAUGH] the tree that you drew isn't quite right though, right? Yeah, it is. because, if you go down the false branch, it means you are less than 20. No, I can be greater than 30. Oh, good one. Yes, I'm very clever, or at least I accidentally got it right. One of the two, it's the same thing. Okay, so there you go.

DECISION TREES: OTHER CONSIDERATIONS

- CONTINUOUS ATTRIBUTES?

e.g. AGE, WEIGHT, DISTANCE

- WHEN DO WE STOP?

→ everything classified correctly!

→ no more attributes!

→ NO OVERFITTING

→ PRUNING

CV?

以下內容(28 和 29 段)要看了 SL2 的 cross validation 後 才能看懂, 日他大爺, 弄到老子半夜三點多, 眼看到看完了, 卻花了好多時間在這段上

28. So, we've answered the thing about continuous attributes. Now, here's another thing. When do we really stop? When we get all the answers right. When all the training examples are in the right category. Class. Right, so the the answer in the algorithm is when everything is classified correctly. That's a pretty good answer, Michael. But what if we have noise in our data? What if it's the case that we have two examples of the same object, the same instance, but they have two different labels? Then this will never be the case. Oh. So, then our algorithm goes into an infinite loop. Which seems like a bad idea. So we could have, we could, we could just say, or we've run out of attributes. [LAUGH] [LAUGH] Or we've run out of attributes. That's one way of doing it. In fact, that, that was, that's going to have to happen at some point, right? That's probably a slightly better answer. Although that doesn't help us in the case where we have continuous attributes and we might ask an infinite number of questions. So we probably need a slightly better criteria. Don't you think? Hm. So, what got us down this path, was thinking about what happens if we have noise. Why would we be worried about having

noise anyway? Noise anyway. Well, I guess the training data might have gotten corrupted a little bit or maybe somebody copied something down wrong. Right, so since that's always a possibility, does it really make sense to trust the data completely, and go all the way to the point where we perfectly classify the training data? But Charles, if we can't trust our data, what can we trust? Well, we can trust our data, but we want to verify. [LAUGH] The whole point is generalization. And if it's possible for us to have a little bit of noise in the data, an error here or there, then we want to have some way to deal to handle that possibility, right? I guess so. So, what will we do? [LAUGH] I mean, we actually have a name for this, right? When you get really, really, really good at classifying your training data, but it doesn't help you to generalize, we have a name for that. Right. That sounds like overfitting. Exactly. We have to worry about overfitting. So you can overfit with the decision tree too? Yeah. What, you don't believe that? No, no, no. I was, I was being naive, I was being, I know that you can overfit a decision tree. [LAUGH] I was just. [LAUGH] Yeah but your [SOUND] is the [SOUND] that you use when you're, when you're like, I don't believe what you just said Charles, but I'm going to go along with it anyway, because I have to get off the phone soon. [LAUGH] Fair enough. I'll try to, I'll try to have a different personality then. [LAUGH] Okay, step one, have a different personality with maximal information gain. Okay, so we don't want to, we don't want to overfit. So we need to come up with some way of overfitting. Now the way you overfit in a decision tree is basically by having a tree that's too big, it's too complicated. All right. Violates Occam's Razor. So, what's a kind of, let's say, modification to something like ID3 to our decision tree algorithm that will help us to avoid overfitting? Well last time we talked about overfitting, we said cross-validation was a good way of dealing with it, which, it allowed us to choose from among the different, say degrees of the polynomial. Right. So maybe we could do something like that? I don't know. Try all the different trees and, see which one has the lowest cross validation error? Maybe there's too many trees. Maybe, but that's a perfectly reasonable thing to do, right? You take out a validation set. You build a decision tree, and you test it on the, on the validation set and you pick whichever one has the lowest error in the validation set, that's one way to avoid it. And then you have, don't have to worry about this question about stopping, you just grow the tree on the training set minus the validation set until it does well on that. And you check it against the crossvalid, you check it against the validation set, and you pick the best one. That's one way of doing it, and that would work perfectly fine. There is another way you can do it that's more efficient. Which is, you do the same idea validation, except that you hold out a set and as you, everytime you decide whether to expand the tree or not, you check to see how this would do so far in the validation set. And if the error is low enough, then you stop expanding the tree. That's one way of doing it. So is there, is there a problem in terms of, I mean if we're expanding the tree depth for search wise, we could be at, you know, we could be looking at one tiny little split on one side of the tree before we even look at any, anything on the other side of the tree. That's a fine point. So how would you fix that? Maybe expand breadth first? Yeah, that would probably do it. Anything else you could think of? Well, so, you could do pruning, right? You could go ahead and do the tree as if you didn't have to worry about over-fitting, and once you have the full tree built, you could then do a kind of, you could do pruning. You could go to the leaves of the tree and say, well, what if I collapse these leaves back up into the tree? How does that create error on my validation set? And if the error is too big, then you don't do it. And if it's very small, then you go ahead and do it. And that should help you with overfitting. So, that whole class of ways of doing it, is called pruning. And there's a whole bunch of different ways you might prune. But pruning, itself, is one way of dealing with overfitting, and giving you a smaller tree. And it's a very simple addition to the standard ID3 algorithm.

From Data Mining with Weka (3.5: Pruning decision trees) video on Youtube:
Pruning is a messy and complicated subject and it's not particularly illuminating.

DECISION TREES: OTHER CONSIDERATIONS

- CONTINUOUS ATTRIBUTES?
e.g. AGE, WEIGHT, DISTANCE
- WHEN DO WE STOP?
PRUNING, WHAT: VOTE
- REGRESSION?
SPLITTING? VARIANCE?
OUTPUT: AVERAGE, LOCAL LINEAR FIT

29. So another consideration we might want to think about with decision trees but you're not going to go into a lot of detail but I think might be worth at least mentioning is the problem of regression. So, so far we've only been doing classification, where the outputs are discrete, but what if we were trying to solve something that looked more like x^2 or $2x + 17$, or some other continuous function. In other words, a regression problem. How would we have to adapt decision trees, to do that? Any ideas Michael? So these are now continuous outputs, not just continuous inputs. Right, maybe the outputs are all continuous, maybe the outputs are discrete, maybe they're a mix of both. Well it certainly seems like the rule of using information gain is going to run into trouble because it's not really clear how you measure information on these continuous values. So, I guess you could measure error some other way. Well we're not, it's not, it's not error right it's trying to measure how mixed up things are? Oh so, maybe something like variance? Cause in a continuous space you could talk about you know, if there's a big spread of, in the values that, that would be measured by the variance. Oh good. So what you really have now is a question about splitting. What's the splitting criteria? Maybe [CROSSTALK] I guess there's also an issue of, of what you do in the leaves. Right. So, what might you do in the leaves? I guess you could do some sort of more standard kind of fitting algorithm. So, like, report the average or, or do some kind of a linear fit. [SOUND] Is any number of things you can do. By the way, that's worth pointing out on the, on the output that if we do pruning like we did before, we have errors, we did actually say when we talked about that how you would report an output. Right? If you don't have a clear answer where everything is labeled true or everything is labeled false, how do you pick? So something like an average would work there. I don't know, I mean, it seems like it depends on what we're trying to measure with the tree. If the tree is, we're trying to get as many right answers as we can, then you probably want to do like a vote in the leaves. Right, which, at least, if the only answer is true or false, that would look more like an average I guess. Right, so you pick, you do a vote. So we do a vote, so we do pruning. We do have to deal with this issue of the output. Somehow, and something like a vote mixing. And here, when you have a regression, then I guess average is a lot

like voting. Yeah, in a continuous phase. Yeah. So either way we're doing a kind of voting. I like that.

DECISION TREES

- REPRESENTATION
- ID3: A TOP DOWN LEARNING ALGORITHM
- EXPRESSIVENESS OF DTs
- BIAS OF ID3
- "BEST" ATTRIBUTES ($GAIN(S, A)$)
- DEALING WITH OVERFITTING

30. Hi Michael, so that covers Decision Trees Excellent. So, since you are the one who is listening, you get to tell me what we have learned today? Well, we learned about the Decision Tree representation, we learned the top down algorithm for inducing a Decision Tree. And we call that ID3 All right. So we got a representation, we got a top down learning algorithm ID3. We learned about the, the expressiveness and the bias. Right. So those are 2 separate things, we learned about the sort of expressiveness. And we learned about the bias of ID3. And we gave one, so is this specific to ID3? We, we looked at one specific way of deciding on splits, which was to do this maximum information game. Right. So we talked about in general, um, what are good attributes or what are best attributes. So, information gain is one way of doing it. As one example. And, [by the way, this notion of best attribute is something we'll end up returning to sometimes explicitly, and sometimes implicitly, throughout the entire course.](#) And lastly, I feel like we talked about the problems with over-fitting and how in the Decision Tree context, [you can prune back the tree to avoid over-fitting.](#) Over-fitting is an issue. Over-fitting is always an issue and we came up with a couple of strategies for dealing with over-fitting in the context of Decision Trees. Okay! So we've learned everything there is to know about Decision Trees, there's nothing else to know. [LAUGH] Somehow I find that hard to believe. Yeah, there's a lot there and the students will get a chance to learn even more as they do the assignments. Cool. Excellent. All right Michael, thank you. Sure, look forward to the next chat.