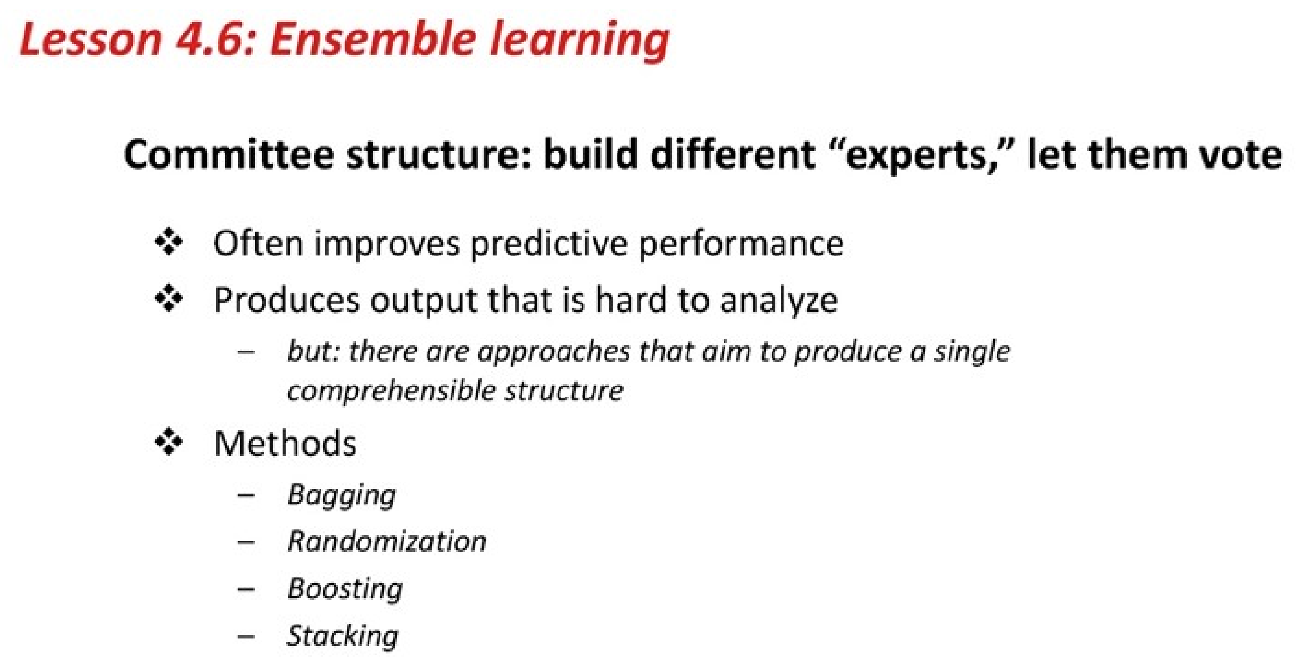
Big Data 課的 note “5. Ensemble Methods” 裡面講了 bagging 的. 這裡沒講 (後面其實講了點的).

下圖來自 Yutube 裡的 Data Mining with Weka (4.6: Ensemble learning):

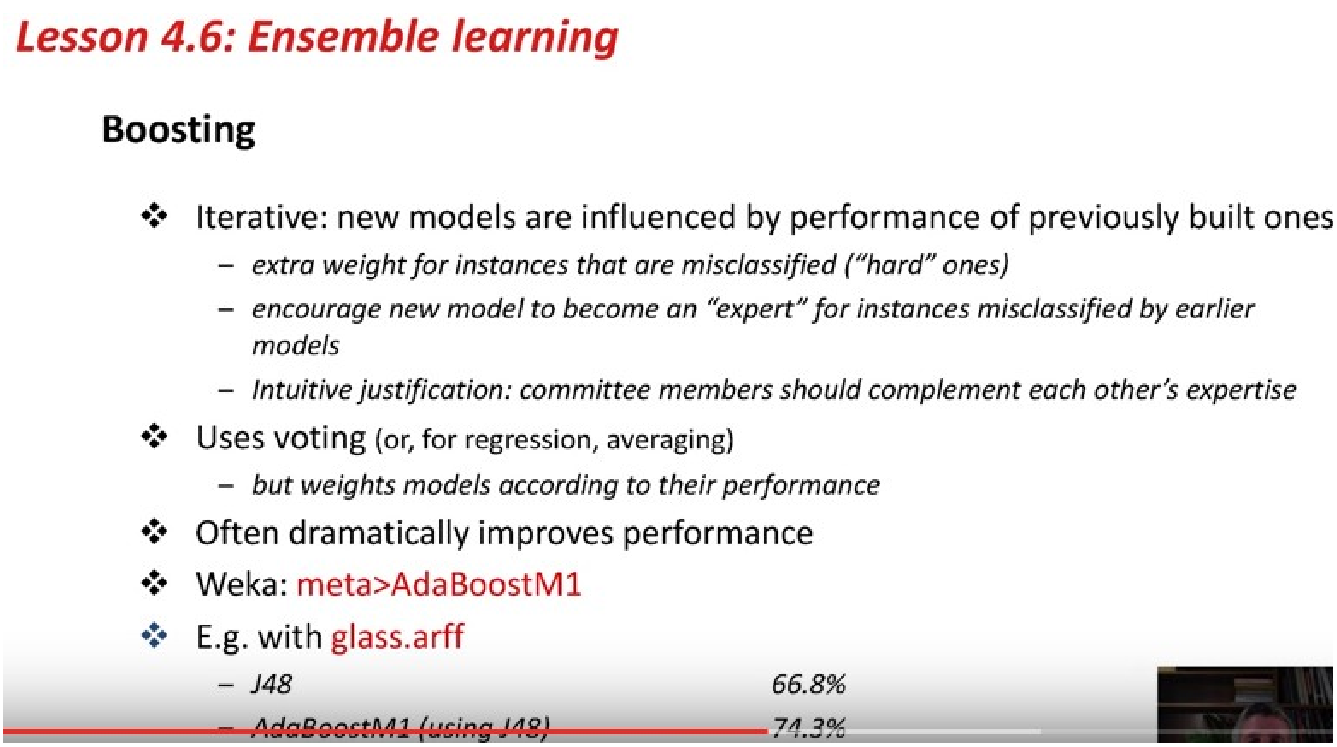


## *Lesson 4.6: Ensemble learning*

### Committee structure: build different “experts,” let them vote

- Often improves predictive performance
- Produces output that is hard to analyze
  - *but: there are approaches that aim to produce a single comprehensible structure*
- Methods
  - *Bagging*
  - *Randomization*
  - *Boosting*
  - *Stacking*

所以 BDT 裡的 forest 中的每一個 tree 就是一個 expert, 然後所有 tree vote. 即每一個 tree 給的結果即 18 段算法中的一個 h_t(x_i) (詳見 18 段).



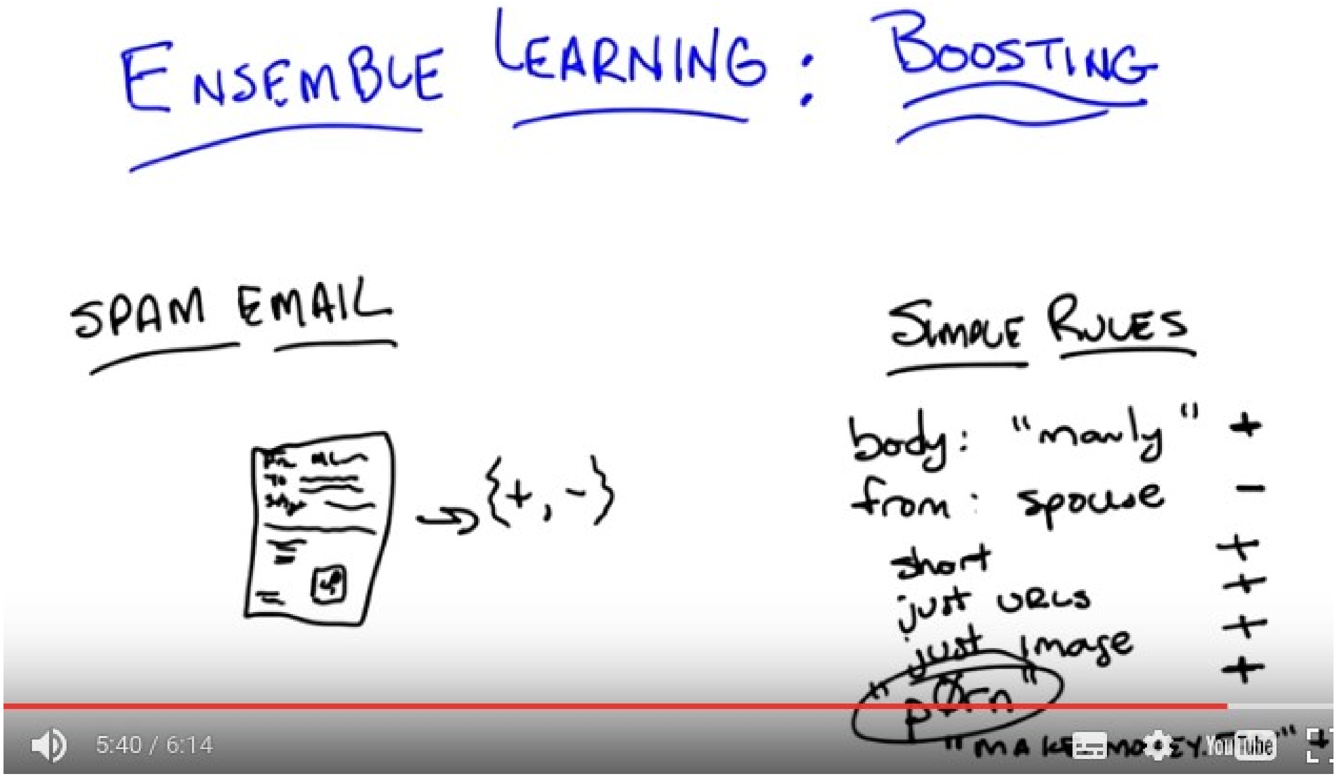
## *Lesson 4.6: Ensemble learning*

### Boosting

- Iterative: new models are influenced by performance of previously built ones
  - *extra weight for instances that are misclassified (“hard” ones)*
  - *encourage new model to become an “expert” for instances misclassified by earlier models*
  - *Intuitive justification: committee members should complement each other’s expertise*
- Uses voting (or, for regression, averaging)
  - *but weights models according to their performance*
- Often dramatically improves performance
- Weka: meta>AdaBoostM1
- E.g. with glass.arff

| | |
|---|---|
| – *J48* | *66.8%* |
| – *AdaBoostM1 (using J48)* | *74.3%* |

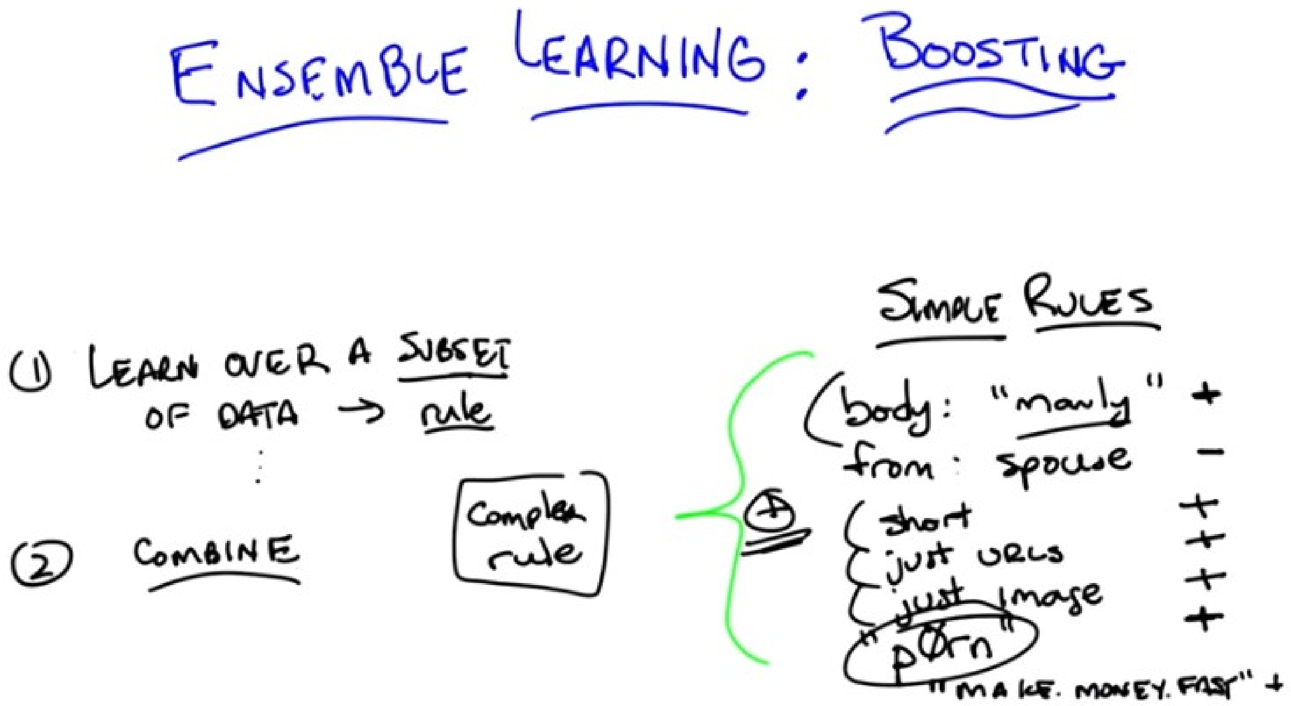
以下是正常的 lecture:



上面每一個 simple rule 應該就是前面說的一個 expert.

1. Hey Charles, how's it goin'? >> It's going pretty well Michael. How's it going with you? >> Good, thank you. >> Good, good, good. Guess what we're going to talk about today? >> Well, reading off this screen, it looks like maybe ensemble learning, and boosting, whatever that is. >> Yes, that's exactly what we're going to talk about. We're going to talk about a class of algorithms called ensemble learners. And I think you will See that they're related to some of the stuff that we've been doing already, and in particular we're going to spend most of our time focusing on, boosting. because boosting is my favorite of the ensemble learning algorithms. So you ready for that? >> Yeah! Let's do it. >> Okay. So, I want to start this out by, going through a little exercise with you. I want you to think about a problem. Okay. And the particular problem I want you to think about is, spam email. >> Mm, I think about that a lot. >> So, normally if we were thinking of this a classification task, right, where we're going to take some email and we're going to decide if it's spam or not. Given what we've talked about so far we would be thinking about using a decision tree or you know neural networks or k and n whatever that means with email. We would be coming up with all of these sort of complicated things. I want to propose an alternative which is going to get us to ensemble learn. OK and here's the alternative. I don't want you to try to think of some complicated Rule that you might come up with that would capture spam email. Instead, I want you to come up with some simple rules that are indicative of spam email. Okay, so let me be specific, Michael. We have this problem with spam email. That is, you you're going to get some email message and you want some computer program To figure out automatically for you if something is a piece of spam or it isn't. And I want you to help write a set of rules that'll help me to figure that out. And I want you to think about simple rules, so can you think of any simple rules that might indicate that something is spam? >> Alright I can, yeah I can think of some simple rules. I don't

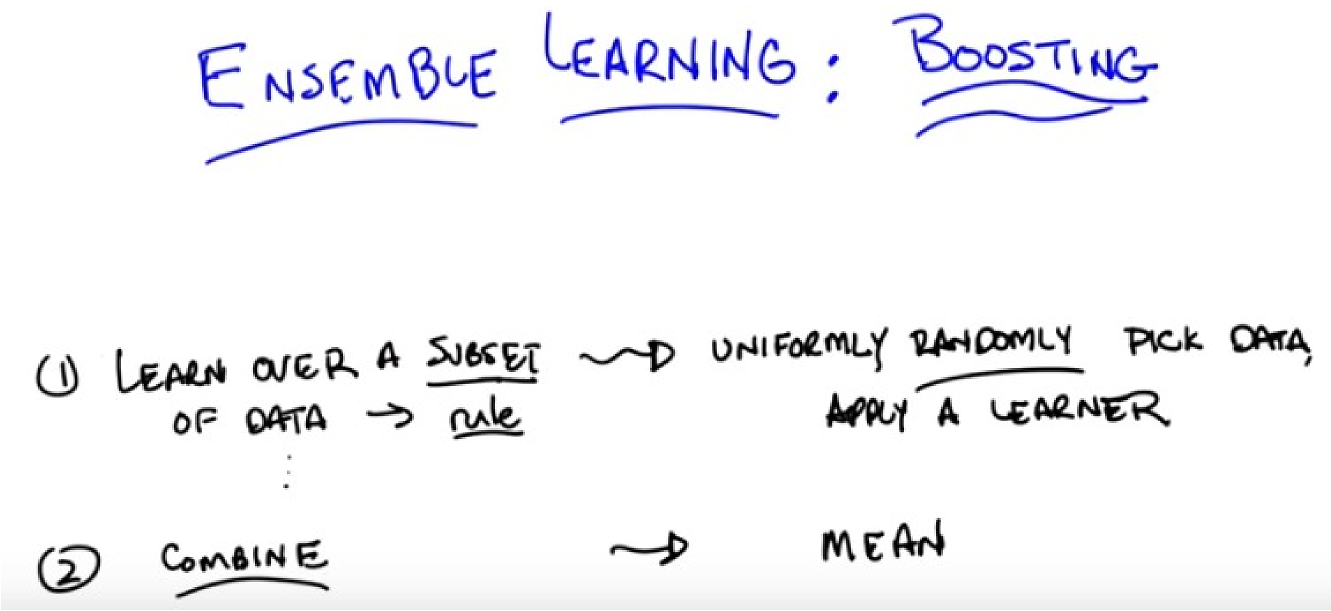
think they would be very good, but they might better than nothing. Like If for example it mentions how manly I am, I, I would be, be willing to believe that was a spam message. So like if the body of the message contains the word manly. >> Okay, I like that. like that when body contains manly. I like that rule, because I often get non-spam messages talking about manly. So I guess one man's spam is another man's normal email. >> [LAUGH] I guess that's true. >> Probably. Any other rules? >> Sure if it, you know if it comes from my spouse it's probably not spam. >> OK, so let's see, from spouse. >> Her name's Lisa. Now we're going to call our spouse. So let's say minus, I'm going to go to plus here, so we know some rules are indicitive of being spam, and some rules are indicitive of not being spam. Okay, anything else? >> Possibly the length of the message. I guess. Like what? >> I don't know. I don't know that this would be very accurate, but I think some of this, some of the spam I get sometimes is very, very short just like the, it's like the URL. Like hey, check out this site, and then there's a URL. >> Hm, I like that. So, we'll just say short. Just contains URLs. Hm, I like those rules. Let's see if we can think of anything else. Oh, how about this one. It's just an image. >> Hm. >> I get a lot of those where it's just an image. >> I see, like in it's it's and if you look at the picture it's all various pharmaceuticals from Canada. >> Exactly. Here's one I get a lot. >> Hm, >> Lots of misspelled words that you end up reading as being a real word. >> Hm. But I don't know how I'd write that as a rule. Or you could just list the words. >> Like rules that, words that have already been modified in that way. I guess so. >> Yeah, kind of a, kind of a black list, a black list of words. >> Okay so, words like, I would say manly, but you were saying prawn. >> Or whatever that says. yeah, so they're and they're tons of these. Right? I mean, another one that's, that's very popular if you're old enough anyway is this one, remember this one? >> Oh, sure that was sometimes a virus, right? >> Yes. Our young, our younger viewers will not know this but this was one of the first big spam messages that would get out there. Make money fast. And there's tons and tons of these. We could come up with a bunch of them. Now, here's something they all kind of have in common, Michael, and you've touched on this all ready. All of them are sort of right. They're useful but no one of them is going to be very good at telling us whether a message has spam on its own. Right. So the word manly is evidence but it's not enough to decide whether something is spam or not. It's from your spouse, it's evidence it's not spam, but sometimes you get messages from your spouse that are in fact spam, because in fact, she didn't actually send them. You know, and so on and so forth. And sometimes she did email from Princes in Nigeria. I didn't. And they're not always spam. I, I actually do, but any case, sometimes people are asking you for money, and maybe that's message you want to ignore, but it isn't necessarily spam. And some people are very interested in getting like this and don't consider it spam, right? >> So, so, okay, so I can see that these would all maybe provide some evidence, but it seems really hard to figure out the right way of combining them all together to I don't know, make a decision. >> Right, this is exactly right. And by the way, if you think about something like decision tree, you could. There's really a sort of similar problem going on there. We can think of each of the nodes in a decision tree as being a very simple rule and the decision tree tells us how to combine them. Right? So, we need to figure out how to do that here and that is the fundamental notion of ensemble learning. >> But wait, isn't, couldn't you also do something similar with something like neural net. Right? Where each of these now becomes a feature and we're just trying to learn ways for combining them all together. So That would kind of satisfy what you were talking about. >> True, I mean I think the the difference here in this case and and I think you're absolutely right but one difference here is that typically with the new network we've already built the network itself and the nodes and we're trying to learn the weights whereas in something like a decision tree you're building up rules as you go along. And typically with ensemble learning you're building up a bunch of rules and combining them together until you got something that's good enough. But you're absolutely right. You could think of [UNKNOWN] networks as being an ensemble of little parts. Sometimes hard to understand, but an ensemble nonetheless.



這裡的 subset of data 應該就是後面講的 misclassified data, 即那些比較難的 data.

2. So, the characteristic of ensemble learning is first this, that you take a bunch of simple rules, all of which kind of make sense and you can see as sort of helping. But on their own, individually, do not give you a good answer. And then you magically combine them in some way to create a more complex rule, that in fact, works really well. And ensemble learning algorithms have a sort of basic form to them, that can be described in just one or two lines. So let me do that and then we can start wondering a little bit how we're going to make that real. So here's the basic form of an ensemble learning algorithm. Basically you learn over a subset of the data, and that generates some kind of a rule. And then you learn over another subset of the data and that generates a different rule. And then you learn over another subset of the data and that generates yet a third rule, and yet a fourth rule, and yet a fifth rule, and so on and so forth. And then eventually you take all of those rules and you combine them into one of these complex rules. So, we might imagine in the email case that I might look at a small subset of email that I know is already spam and discover that the word manly shows up in all of them and therefore pick that up as a rule. That's going to be good at that subset of mail, but not necessarily be good at the other subset of mail. And do the same thing and discover that a lot of the spam mails are in fact short or a lot of them are just images or just urls and so on and so forth. And that's how I learn these rules by looking at different subsets. Which is why you end up with rules that are very good at a small set of the data, but aren't necessarily good at a large set of the data. And then after you've collected these rules, you combine them in some way, and there you go. And that's really the beginning and the end of ensemble learning. >> So wait. So, when you say manly was in a lot of the positive examples. Do you mean like it distinguishes the positive and the negative example? So it should also not be in the negative examples. >> That's right. That's exactly right. So think of this as any other classification learning problem that you would have where you're trying to come up with some way to distinguish between the positives and the negatives. >> And, and now why does it, why are we are looking at subsets of the data? I don't understand why we can't just look at all, the whole data. >> Well, if we look at all of the data, then it's going to be hard to come up with these, these simple rules. That's

the basic answer. Actually, ask me that question a little bit later, when we talk about over fitting, and I think I'll have a good answer for you. Okay, so here we go Michael. This is Ensemble Learning. You learn over a subset of the data over and over again picking up new rules and then you combine them and you're done.



ENSEMBLE LEARNING : BOOSTING

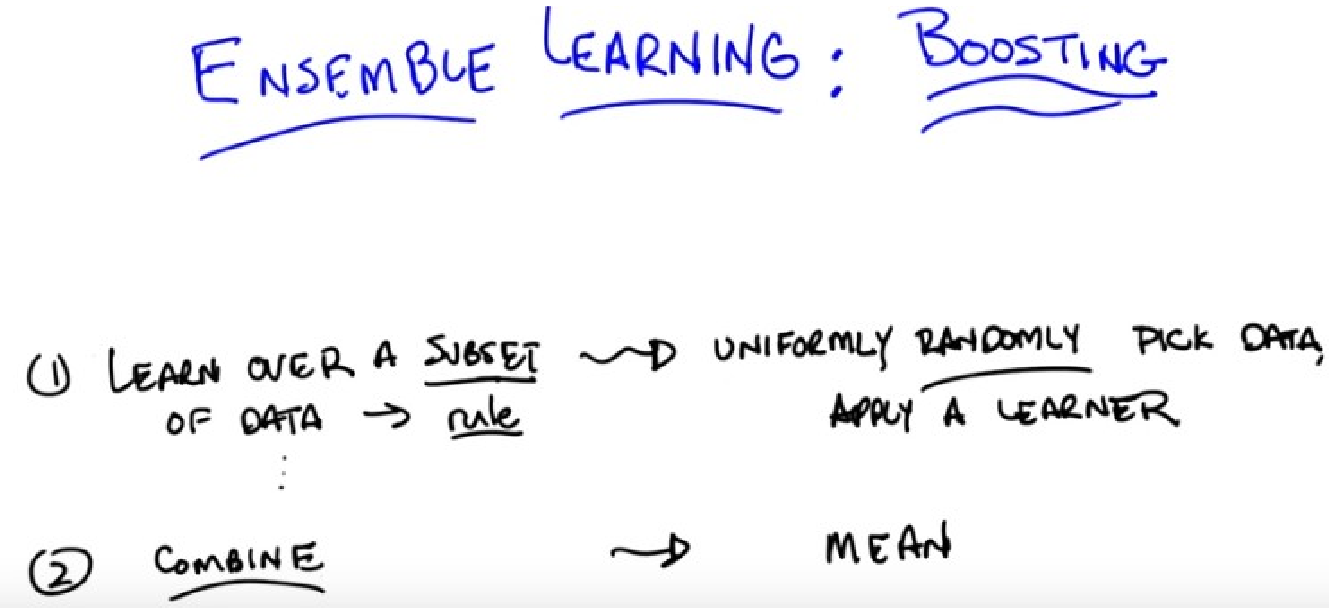
(1) LEARN OVER A SUBSET OF DATA → rule ⤳ UNIFORMLY RANDOMLY PICK DATA, APPLY A LEARNER

⋮

(2) COMBINE ⤳ MEAN

3. Here's the Ensemble Learning algorithm. We're done, Michael, we're done with the entire lesson.  We don't have to do anything else anymore. We know that we're supposed to look over subset of data, pick up rules, and then combine them. So, what else do you need to know in order to write your first Ensemble Learning algorithm?  >> So, I'm already kind of uncomfortable with this notion of combine, right? So, like, I can think of lots of really dumb ways to combine things. Like, choose one at random or, you know, I don't know, add them all up and divide by π. I mean so,so presumably there's gotta be some intelligence in how this combination is taking place >> Yes, you would think so, but your not at all bothered about how you pick a subset?  >> Oh ,I was imaging you meant random subsets.  >> Oh, so you'd automated assumption about how we were going to pick a subset. You just [CROSSTALK] werent sure how to combine them. Well actually lets explore that for a minute. Here's kind of the dumbest thing you can imagine doing. That turns out to work out pretty well.  We're going to pick subsets, by, I'm going to say uniformly. Just to be specific about it. So ,we're going to do the dumbest thing that we can think of, or one one of the dumbest things you could think of. Or maybe ,we should say simplest and not dumbest so as not to, to, to make a value judgment. That you can think of doing which would be to just uniformly randomly Choose among some of the data, and say that's the data I'm going to look at, and then I'm going to apply some learning algorithm to it. Is that what you were thinking of Micheal?  >> Yeah.  >> Okay, so just pick a subset of the data, apply a learner to it. I'll get some hypothesis out, I'll get some rule out. And now I'm going to combine them, so since were being simple. Why don't we try doing something simple for combining. Let's imagine, Michael, that we're doing a regression. What's kind of the simplest thing you could do if you have ten different rules which tell you, how you should be predicting some new data point? What's the simplest thing you could imagine doing with it?  >> So, okay, so each of them spits out a number. I guess if we kind of equally believe in each of them, a reasonable thing to do would be to average.  >> Great. So, a very simple way of combining, in the case of regression, would be to average them. We'll simply take the mean. And by the way, why wouldn't we equally believe in each of them. Each one of them learned over a random

subset of the data. You have no reason.  >> Well.  >> To believe one's better than the other.  >> There's a couple of reasons. One ,it could be a bad random subset. I don't know how I would measure that.  >> I could be a good random subset.  >> Yeah. Then we'd want, we'd want that to count more in the mean. But, but I guess what I was thinking more in terms of maybe for some of the subsets you know, it gets more error than others or it uses more complex rule than others or something.  >> I could imagine that. Actually maybe we can explore how this sort of idea might go wrong. Let's, let's do that. Maybe we can do that with the quiz.  You like quizzes, right?  >> They're important.

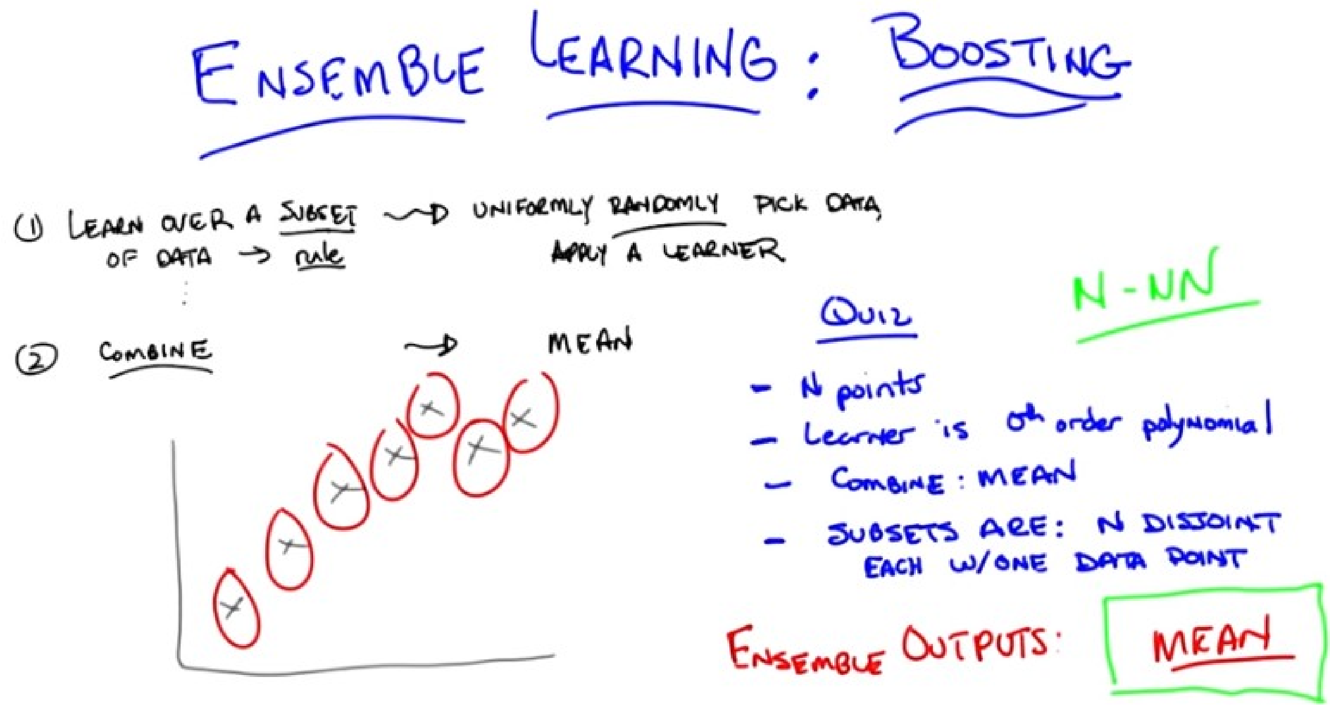


ENSEMBLE LEARNING : BOOSTING

(1) LEARN OVER A SUBSET OF DATA → rule ⇝ UNIFORMLY RANDOMLY PICK DATA, APPLY A LEARNER

⋮

(2) COMBINE ⇝ MEAN

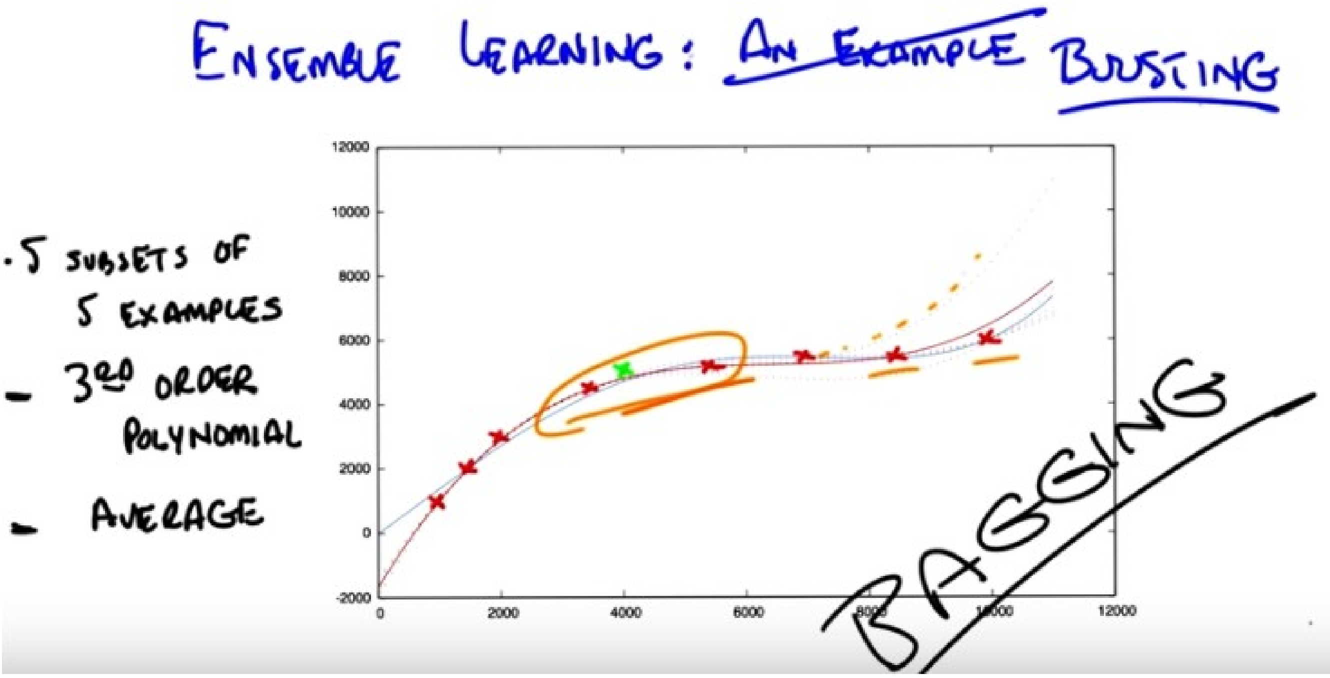

ENSEMBLE LEARNING : BOOSTING
(1) LEARN OVER A SUBSET OF DATA → rule
UNIFORMLY RANDOMLY PICK DATA
APPLY A LEARNER
(2) COMBINE → MEAN
QUIZ
N-NN
- N points
- Learner is 0th order polynomial
- COMBINE : MEAN
- SUBSETS ARE: N DISJOINT EACH W/ONE DATA POINT
ENSEMBLE OUTPUTS: MEAN

上圖右邊是 Learner is 0th order polynomial

4. Okay, so here's a quiz for you Micheal, here's the setup, you ready? You've got in data points. The learner that you're going to use over your subsets is a zeroth order polynomial. The way you're going to combine the output of the learners is by averaging them. So, it's just what we've been talking about so far, and your subsets are going to be constructed in the following way. You uniformly randomly picked them and you ended up with N different subsets or disjoint, and each one has a single point in it, that happens to be one of the data points. >> Okay i think i get that >> Right, so if you look over here on your left you got a graph of some data points and this is one subset This is another subset, that's another subset, that's another subset, that's another subset, that's another subset, that's another subset, got it. >> Yeah, now what do you want to know about it. >> Now what I want to know is when you do your ensemble learning you learn all these different, you learn all these different rules and then you combine them ,what is the output going to be? What does the ensemble output? >> And you want a number? >> I want a description and if the answers a number that's a perfectly fine description. But I'll give you a hint, it's a short description. >> A short description of the answer. Okay, I'll think about it. >> Alright.

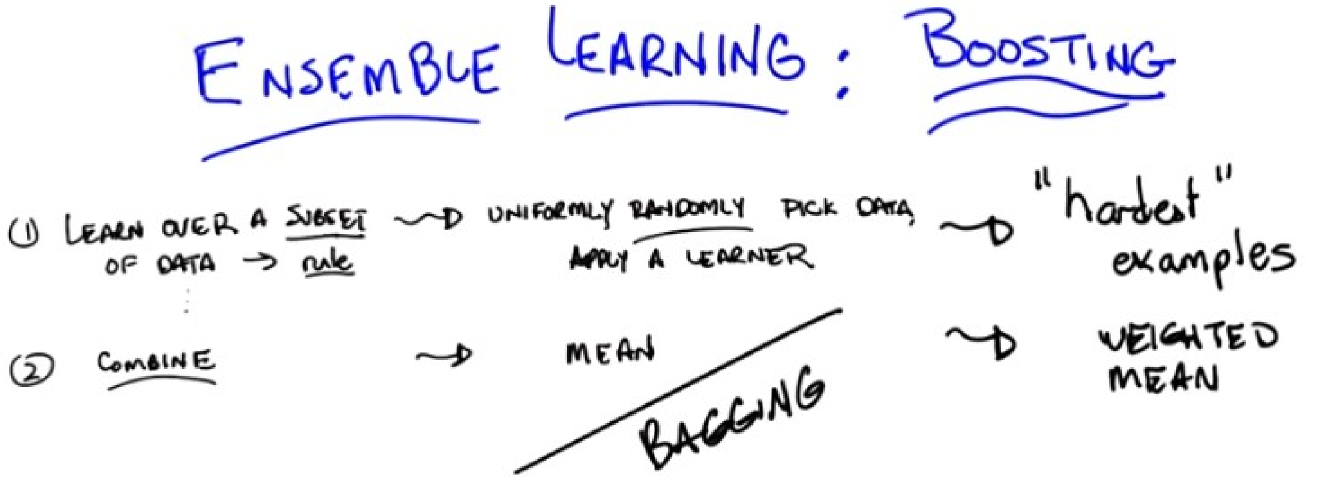
5. Okay Michael have you thought about it? Do you know what the answer is? >> Yeah. I think, you know, you asked it in a funny way, but I think, what you're asking maybe was pretty simple. So let, let me, let me see if I can talk it through. So ,we've got n data points, each learner is a zeroth order polynomial. So you, you said the ensemble rule is that you learn over a subset, a zeroth order polynomial is just (no period) Well, we said that the thing that minimizes the average. Sorry, that minimizes the expected error, or the squared error [INAUDIBLE] it's just the average. So, if the sets are indistinct sets, with one data point each, then each, of the individual learners, is just going to learn the average. Then they get, not the average sorry. The actual output value of each individual point is the average, and then the combining algorithm, to combine all the pieces of the ensemble into one answer, combines with the mean. So ,it's going to combine the mean of those each of which is the data point, so it's the mean of the data points. So, the ensemble outputs, I don't know, I'd say average or mean? >> Yes. >> Or zeroth [INAUDIBLE] polynomial of the data set, or, you know, one node decision tree,

or ,uh. >> A constant? Which happens to be the mean of the data. Haven't we seen this before? >> It seems to come up a lot, when we are outputting very simple hypotheses. >> Right. And the last time we did this, if I recall correctly, this is what happens ,if you do an unweighted average with k and n where k is equal to n. >> Oh, right. Like, like, right. And N-NN. >> N-NN. >> Mm. >> Mm, so we should probably do something a little smarter than this then. And, I thought that we might look at some of the housing data, because, no one's started looking at the housing data yet. [LAUGH] Okay, so let's look at that right quick and see if we can figure out how this works. And then see if we can do something a little bit better, even better than that. Okay?



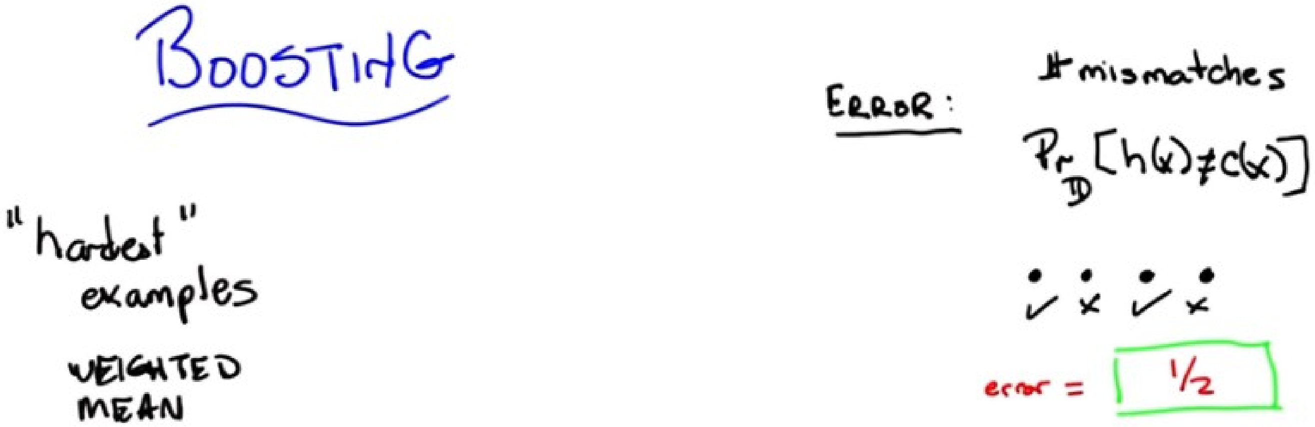
6. Alright, Michael, so, here's what you have before you. You have the same housing data that we've looked at a couple of times before. I've, for the sake of readability, I've drawn over, some of the data points so that they're easier to see. This is exactly the data, that we've always had. Okay? >> Okay. >> Now, you'll notice that I marked one of them as green, because here's what we're going to do. I'm going to take the housing data you've got, I'm going to do some ensemble learning on it. And I'm going to hold out the green data point. Okay? So of the 9 data points (包括那個 green 的), you're only going to learn on 8 of them (不包括那個 green 的). And I'm going to add that green data point as my test example and see how it does. Okay? >> Okay. So that sounds like, cross validation. >> It does. This is a cross validation. Or you could just say, I just put my training set and my test set on the same slide. >> Okay. >> Okay, Michael, so the first thing I'm going to do is I'm going to pick a random subset of these points (from the 8 points). And just for the sake of the example, I'm going to pick five points randomly. And I'm going to do that five times. So I'm going to have five subsets of five examples. And by the way, I'm going to choose those randomly, and I'm going to choose them with replacement. So we're not going to end up in the situation we ended up just a couple of minutes ago where we never go to see the same

data point twice. Okay? >> Yeah. >> Alright. So 5 subsets of 5 examples, and then I'm going to learn a third order polynomial. And I'm going to take those 3rd order polynomials. I'm just going to learn on that subset, and then I'm going to combine them by averaging. Want to see what we get? >> Oh, yeah, sure. >> So here's what you get, Michael. Here I'm showing you a plot over those same points, with the five different 3rd order polynomials (仔細看, 就有很多條虛線, 每條虛線就是一個那 5 個 subsets 中的一個 subset 弄出來的). Can you see them? >> Yeah. They're, right. There's like a bunch of wispy hairs. >> Just like most third order polynomials. And as you can see they're, they're kind of you stare at them and you see their kind of similar. But some of them they veer(改變方向或路線) off a little bit because they're looking at different data points. One of them(即那 5 個 subsets 中的某個 subset 弄出來的線) actually very hard to see because it's only one like this. Actually veers off like this(那條黃色的尾部向上的 dashed curve) because just, purely randomly, it never got to see the two final points. >> I see. But they all, but they all seem to be pretty much in agreement, like between points three and four. There's a lot of consistency there. >> Right. Because just picking five of the subsets you seem to be able to either get things on the end, or you get things in the middle. And maybe one or two things on the end it sort of works out. Even the one that doesn't see the, the last two points still got to see a bunch of first ones and get this part of this space fairly right. >> Cool. >> Okay. So the question now becomes how good is the average of these compared to something we might have learned over the entire data set? And here's what we get when do that. So what you're looking at now Michael, is the red line(紅色的實線), is the average of all of those five third order polynomials. And the blue line(藍色的實線), is the fourth order polynomial that we learned when we did this with simple regression, a couple of lessons back. >> Okay. >> And you actually see them pretty close. >> Why is one of them a fourth order, and one a third order? >> Well what I wanted to do is try a simpler set of hypothesis, than we were doing, than when we were doing full blown regression. So third order simpler than fourth order. So, I thought we'd combine a bunch of simpler rules. Then the one we had used before and see how well it does. >> You want to know how well it does? >> I would! >> Well it turns out that on this data set and I did this many, many, many times just to see what would happen with many different random subsets. It typically is the case that the blue line always does better on the training set, the red points, than the red line does. But the red line almost always does better on the green point on the test set or the validation set. >> Interesting. >> That is kind of interesting. So wait, so let me get this straight. It seems sort of magical. So, so it learns an average of third degree polynomials, third order polynomials, which is itself a third order polynomial. But you're saying it does better by doing this kind of trick than just learning a third order polynomial directly. >> Yeah, why might you think that might be? I have a guess, you tell me what you think. >> Wow, so well, I mean, you know, the danger is often over fitting, over fitting is like the scary possibility. And so maybe by, by kind of mixing the data up in this way and focusing on different subsets of it. I don't know. Somehow manages to find the important structure as opposed to getting misled by any of the individual datapoints. >> Yeah. That's the basic idea. It's kind of the same thing, at least that's what I, I think that's a good answer. It's basically the same kind of argument you make for cross validation. Alright. You take a random bunch of subsets. You don't get trapped by one or two points that happen to be wrong because they happen to be wrong because of noise or whatever. And you sort of average out all of the variances and the differences. Hm. And often times it works. And in practice this particular technique. Ensemble learning does quite well in getting rid of overfitting. >> And what is this called? >> So, this particular version, where you take a random subset and you combine by the mean, it's called **bagging**. >> And I guess the bags are the random subsets? >> Sure. >> [LAUGH] That's how I'm going to think of it. >> That's how I'm going to think of it. It also has another name which is called bootstrap(靴帶) aggregation. So I guess the different subsets are the boots. >> [LAUGH] No,no, no, no bootstrap usually refers to pulling yourself up by your bootstraps. >> Yeah, I like my, I like my answer better. So, each of the subsets are the boots and the averaging is the strap. And there you go. So, regardless of whether you call it bootstrap aggregation or you call it bagging, you'll notice it's not what I said we were going to talk about during today's discussion. I said we were going to talk about boosting. So we're talking about bagging but we're going to talk about boosting. The reason I wanted to talk about bagging is because it's really the simplest thing you can think of and it actually works remarkably well. But there are a couple of things that are wrong with it, or a couple of things you might imagine you might do better. That might address some of the issues and we're going to see all of those when we talk about boosting right now.


ENSEMBLE LEARNING : BOOSTING
(1) LEARN OVER A SUBSET OF DATA → rule
UNIFORMLY RANDOMLY PICK DATA, APPLY A LEARNER
"hardest" examples
(2) COMBINE
MEAN
BAGGING
WEIGHTED MEAN

7. Okay so, let's go back and look at our two questions (即如何找 subset, 如何 combine) we were trying to answer. And so far we've answer the first one, learn over a subset of data, defined a rule by choosing that subset uniformally randomly and applying some learning algorithm. And we answered the second question, which is how do you combine all of those rules of thumbs by saying, you simply average them. And that gave us, bagging. So Michael, I'm going to suggest an alternative to at least the first one. And leave open the second one for a moment. That's going to get us to what we're supposed to be talking about today, which is boosting. So let me throw and idea at you and you tell me if you think it's a good one. So rather than choosing uniformly randomly over the data, we should try to take advantage of what we are learning as we go along, and instead of focusing just kind of randomly, we should pick the examples that we are not good at. So what do i mean by that? What I mean by that is, we should pick a subset based upon whether the examples in that subset are hard. So what do you think of that? >> Well, I guess it depends on how we think about hard, right so it could be that it's hard because some, in some absolute sense right, or could be that it is hard relative to you know, if we were to stop now how well we do Yeah, and I mean the latter. >> Oh. Okay. Alright. Well that I feel like that makes a lot of sense. I mean, certainly when I'm you know, trying to learn a new skill, I'll spend most of my energy on the stuff that I kind, that I'm kind of on the edge of being able to do, not the stuff that I've already mastered. It can be a little dispiriting. But it, but it I think it, I make faster progress that way. >> Right and if you, if you go back to the example that we, we started with, with spam right? If you come up with a and you see it does a very good on some of the data, some of the mail examples, but doesn't do a good job on the other. Why would you spend your time trying to come up with more rules that do well on the email messages you already know how to classify? You should be focusing on the ones you don't know how to classify. And that's the basic idea here between, the basic idea hearer behind boosting and finding the hardest examples. >> Cool. >> Okay. So that answers the first one. We're going to look at the hardest examples, and I'm going to define for you exactly what that means. I'm going to have to introduce at least one technical definition. But ,uh, I want to make certain you got that. And the second one, the combining, well that's a difficult and sort of complicated thing, but at a high level, I can explain it pretty easily by saying we are going to still stick with the mean. >> Okay. >> We're voting, except this time,this time we are going to do weighted mean. Now why do we want to do

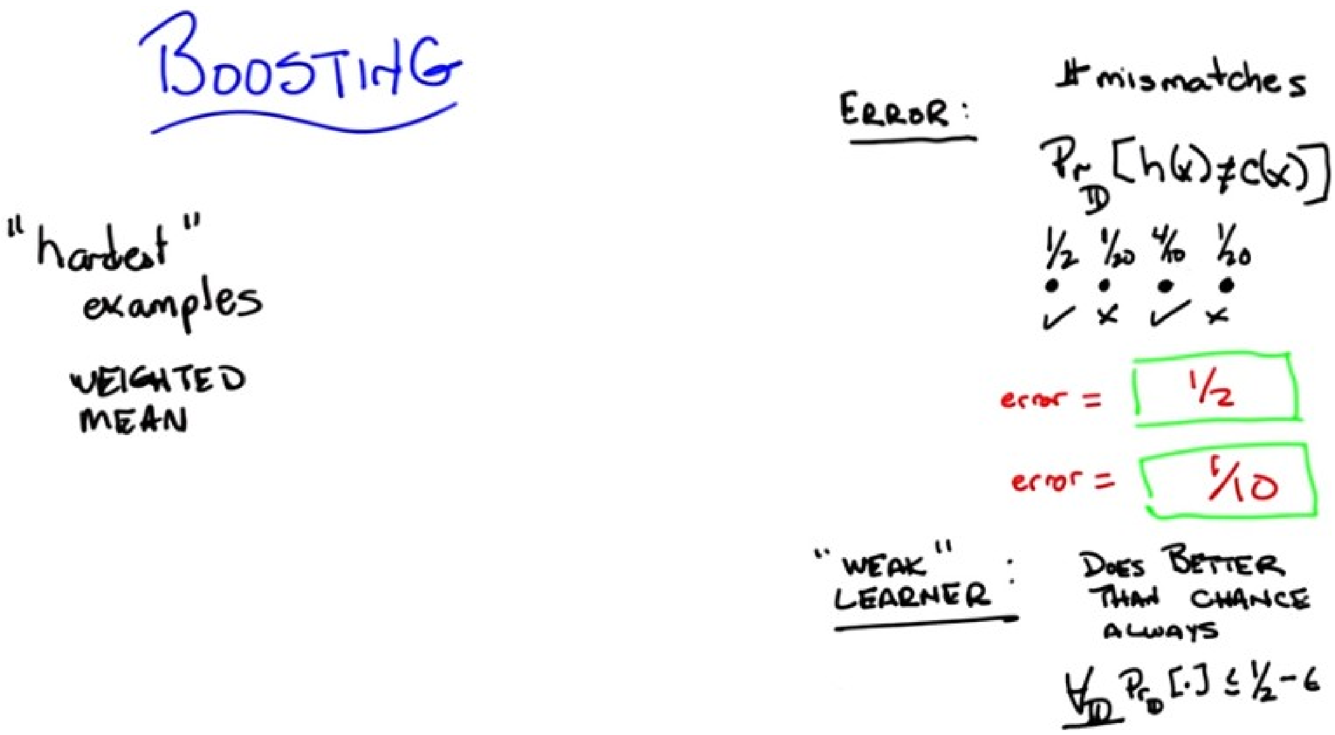
weighted mean? Well I have to tell you exactly how we are going to weight it but the basic idea is to avoid the certain situations That we came across when we looked at the data before, when taking an average over a bunch of points that are spread out, this gives you an average or a constant that doesn't give you a lot of information about the space. So we're going to weight it by something, and it's going to turn out the way we choose to weight it will be very important. But just keep in your head for now that we're going to try to do some sort of weighted average. Some kind of weighted voting. Okay? >> Sure. One of the things that's scaring me at the moment though is this, like I have this fear that by focusing on the hardest questions, and then, and then sort of mastering those, what's to keep the learner from starting to kind of lose track of the ones it has already mastered? Like how, why does it not thrash back and forth? >> So that's going to be the trick. Behind the, the particular way that we do weighting. >> Okay >> So I will show you that in a moment, and it's going to require two slightly technical definitions, that we have been kind of skirting around, this entire conversation. Okay? >> Sure.



8. Alright so, the, the whole goal of what we're going to add for boosting here is we're going to, we're going to expand on this notion of hardest examples and weighted mean. But before I can do that, I'm going to have to define a couple of terms. Okay. And you let me know Michael if, if these terms make sense. So, here's the first one. The first one is error. So how have we been defining error so far? >> Usually we take the square difference between the correct labels and the, what's produced by our classifier or regression algorithm. >> That's true. That is how we've been using error when we're thinking about regression error. How about, a notion of accuracy. About how good we are at, say, classifying examples. So let's, let's stick with classification formulas. >> Well, that would be the same as squared areas, except that it's not really meeting the whole powers [INAUDIBLE] area. That is to say, if the outputs are zeroes and ones, the squared area is just whether or not there's a mismatch. So it could just be the total number of wrong answers. >> Right. So, what we've been doing so far is counting mismatches. I like that word, mismatches. And we might call an error raid or an error percentage as the total number of mismatches over the total number of examples. And that tells us whether we're at 85% or 92%, or, or whatever, right? So that's what we've been doing so far. But implicit in that, Michael, is the idea that every single example is equally as important. So, that's not always the case. Now you might remember from the first talk that we had. We talked about distributions over examples. We said that, you know, learning only happens if you're training set has the same distribution as your future testing set. And if it doesn't, then all bets are off. And it's very difficult to talk about induction or learning. That notion of distribution is implicit in everything that we've been doing so far, and we haven't really been taking into account when we've been talking about error. So here's another definition of error and you tell me if you think it makes sense, given what we just said.

So, this is my definition of error. So the subscript D, stands for distribution(distribution 的意思見下面的第 9 段). So we don't know how new examples are being drawn, but however they're being drawn they're being drawn from some distribution, and I'm just going to call that distribution" D", okay? >> mhm Right. So h is our old friend the hypothesis. That's the specific hypothesis that our learner has output. That's what we think is the true concept. And C is whatever the true underlying concept is. So I'm going to define error as the probability, given the underlined distribution that I will disagree with the true concept on some particular instance x. Does that make sense for you? >> Yeah but I'm not seeing why that's different from number of mismatches in the sense that if we count mismatches on a sample drawn from D, which is how we would get our testing set anyway. Then I would think that would be you know if it's large enough a pretty good approximation of this value. >> So here Michael, let me give you a specific example. I'm going to draw four, four possible values of x. And when I say I'm going to draw four possible values of X, I mean I'm just going to put four dots on the the screen. >> Hm. >> Okay? And then I'm going to tell you this particular learner output a hypothesis. Output you know, a, a potential function that ends up getting the first one and the third one right, but gets the second and the fourth one wrong. So what's the error here? >> Mm. >> So let's just make sure that, that everybody's with us. Let's do this as a quiz. >> Okay, so let's ask the students what they think. So here's the question again. You've output some hypothesis over the four possible values of x, and it turns out that you get the first and the third one right, and you get the second and the fourth one wrong. If I look at it like this, what's the error rate?

9. Okay, Michael what's your answer? >> It looks like, half of them are right and half of them are wrong. So, the number of mismatches, is, two out of four or a half. >> Right, that is exactly the right answer ,because ,you got half of them right and half of them wrong. But ,it assumes ,that your likely to see all four of the examples, equally often. So, what if I told you, that, that's not in fact the case. So, here's another example of error for you. What if I told you that each of the points, is, likely to be seen ,uh, in different proportions and in fact in these particular proportions. Distribution 就是這個意思: So you're going to see the first one, half the time. You're going to see the second one, 1/20 at a time. You're also going to see the fourth one, 1/20 at a time and you'll see the third one 4/10 of the time. Alright, so you got it Michael? One half, one 20th, four tenths and one twentieth. >> Got it.

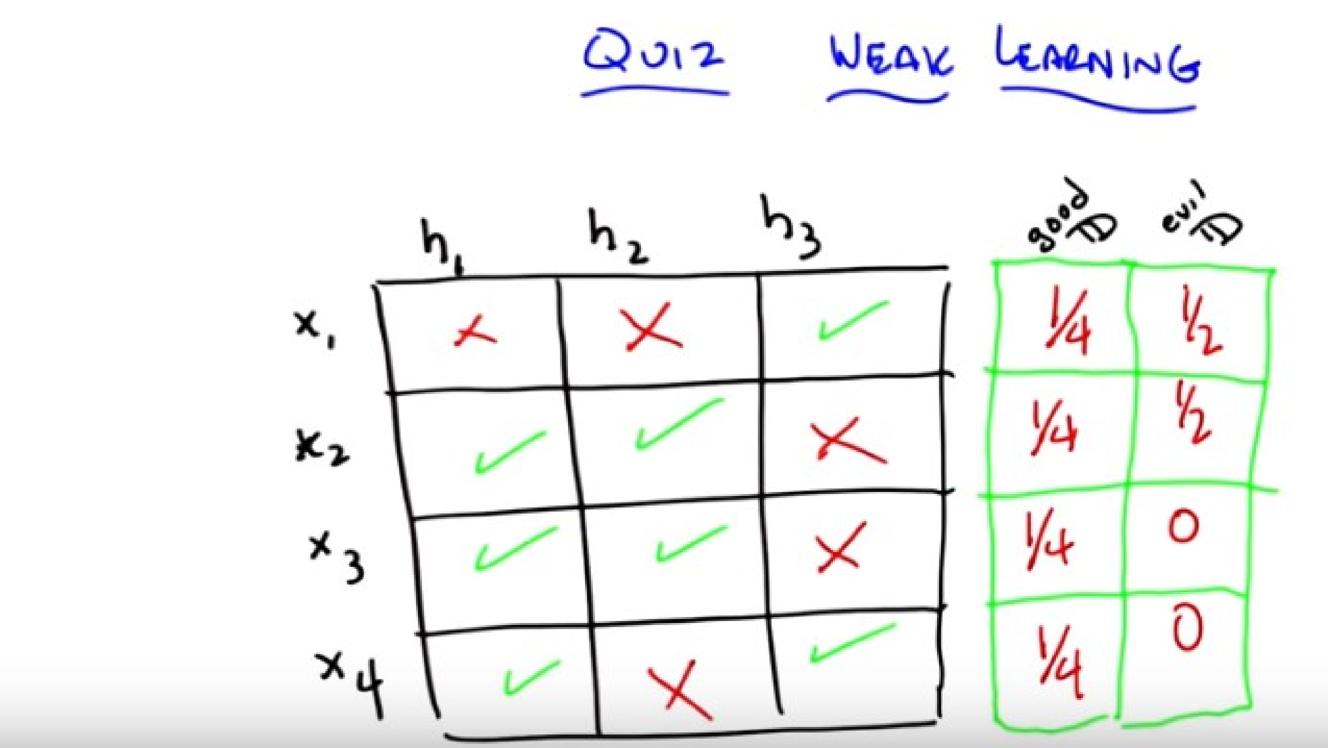


右下角是 1/2 - σ
error = (1/20 + 1/20) / (1/2 + 1/20 + 4/10 + 1/20) = 1/10.
1/2 表示這個東西出現在大街上的概率為 1/2. Distribution 就是那個 出現率 的分佈, 即那個(1/2, 1/20, 4/10, 1/20), think of 概率統計中的 概率分佈函數.

10. Okay. So, now I have a different question for you. Actually, I have the same question for you, which is, what is the error rate now. Go.

11. Okay, Michael what's the answer? >> Well, it's still a half. But I guess we, we really should take into consideration those probability. So the number of mismatches they have, but the actual number of errors, the expected number of errors is like well, a 20th plus 20th, so like a 10th. So it's 90% correct, 10% error. >> Right. That's exactly right, so, what's important to see here is that even though you may get many examples wrong, in some sense some examples are more important than others. Because some are very rare. And if you think of error, or the sort of mistakes that you're making, not as the number of distinct mistakes you can make, but rather the amount of time you will be wrong (此話說得好), or the amount of time you'll make a mistake, then you can begin to see that it's important to think about the underlying distribution of examples that. You see. You buy that? >> Yeah. >> Okay, so, that notion of error turns out to be very important for boositng because in the end, boosting is going to use this trick of distributions in order to define what hardest is. Since we are going to have learning algorithms that do a pretty god job of learning on a bunch of examples. We're going to pass along to them a distribution over the examples, which is another way of saying, which examples are important to learn, versus which examples are not as important to learn. And that's where the hardest notion is going to come in. So, every time we see a bunch of examples, we're going to try to make the harder ones more important to get right. Than the ones that we already know how to solve. And I'll, I'll describe in a minute exactly how that's done. >> But isn't it the case that this distribution doesn't really matter? You should just get them all right. >> Sure. But now it's a question of how you're going to get them all right. Which brings me to my second definition I want to make. And that second definition is a weak learner. So there's this idea of a learning algorithm, which is what we mean by a learner here. As

being weak. And that definition's actually fairly straightforward so straightforward in fact that you can sort of forget that it's really important. And all a **weak learners** is, is a learner that no matter what the distribution is over your data, will do better than chance when it tries to learn labels on that data. So what does does better than chance actually mean? Well what it means is, that no matter what the distribution over the data is, you're always going to have an error rate that's less than a half. So that means sort of as a formalism, is written down here below. That for all D, that is to say no matter what the distribution is, your learning algorithm We'll have an expected error. That is the probability that it will disagree with it through actual concept if you draw a single sample that is less than or equal to one half minus Epsilon. Now σ is a term that you end up seeing a lot in mathematical proofs and particularly ones involving machine learning. And σ just means a really, really small number somewhere between a little bigger than 0 and certainly much smaller than 1. So, here what this means technically is that you're bounded away from 1 1/2. Which another way of thinking about that is you always get some information from the learner. The learner's always able to learn something. Chance would be the case where your probability is 1/2 and you actually learn nothing at all which kind of ties us back into the notion of information gained way back when with decision trees. So does that all make sense Michael? >> I'm not sure that I get this right. Let's, maybe we can do a quiz and just kind of nail down some of the questions that I've got. >> Okay, sure. You got an idea for a quiz? >> Sure.



QUIZ WEAK LEARNING

| | $h_1$ | $h_2$ | $h_3$ | good D | evil D |
|---|---|---|---|---|---|
| $x_1$ | ✗ | ✗ | ✓ | 1/4 | 1/2 |
| $x_2$ | ✓ | ✓ | ✗ | 1/4 | 1/2 |
| $x_3$ | ✓ | ✓ | ✗ | 1/4 | 0 |
| $x_4$ | ✓ | ✗ | ✓ | 1/4 | 0 |

先看第 14 段中我對(x_i, y_i) pairs 和 h 的理解, 上圖中的一個 X_i 即相當於一個(x_i, y_i) pair.
goodD 情況下, h1 就是那個要找的, 它的 error 為 1/4, 小於 half. 但它不是 weak learner, 因為 evilD 情況下, h1(以及 h2, h3)的 error 都等於 1/2, 而不是小於 1/2.
由此可知, weak learner 是一個 hypothesis (14 段已驗證).

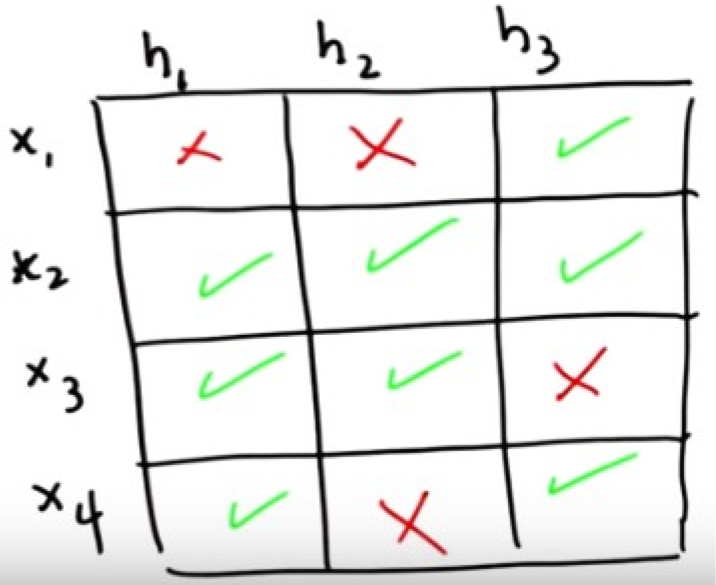
12. Okay Michael so you, let's make certain that you really grasp this concept of weak learning okay? >> Mm-hm. >> So, here's a little quiz that I put together to test your knowledge. So, here's the, here's the deal. I've got a little matrix here, it's a little table, and across the top are three different hypotheses

(h1, h2, h3). So, hypothesis one, hypothesis two, and hypothesis three. So your entire hypothesis base consists only of these three hypothesis, hypotheses. Got it?  >> Got it.  >> Okay, your entire instance space consists entirely of only four examples: x1, x2, x3, and x4. Got it?  >> Got it.  >> I have an X in a square, if that particular hypothesis does not get the correct label for that particular instance, and I have a green check mark if that particular hypothesis does in fact get the right label for that example. So, in this case hypothesis one gets examples 2, 3, and 4 correct. But gets example one wrong, while hypothesis three gets one in four correct, but two and three incorrect.  >> I see. So, there's no hypothesis that gets everything right.  >> Right.  >> So does that mean that we don't have a weak learner, because then there's some distributions for which any given hypothesis is going to get things wrong.  >> Maybe. Maybe not. Let's see. Here's what I want you to do. I want you to come up with the distribution over the 4 different examples, such that a learning algorithm that has to choose between one of those 3 hypotheses will in fact be able to find one that does better than chance. That is, has an expected error less than half [注 1].  >> Okay. Then if you can do that, I want you to see if you can find a distribution, which might not exist, such that if you have that distribution over the four examples, a learning algorithm that only looked at h1, h2 and h3 would not be able to return one of them that has an expected error less than half [注 2].  >> So greater than half in this case would mean three out of four, correct?  Oh no, no. Oh, you're using, you want to use that definition that you, that actually took into consideration the distribution.  >> Exactly. That's the whole point. If you, if you always need to have some distribution over your examples to really know what your expected error is.  >> Alright. And if there is no such evil distribution, should I just fill in zeroes in all those boxes? Yes, all zeros means no such distribution. You can do it in either case. So if you put in all zeros you're saying no such distribution exists. But otherwise it should add up to one down each of the columns.  >> It had better add up to one.  >> Alright, I think I can give that a shot. Okay, go.

[注 1]: 原文為 has an expected error greater than half. 但由上一段知, better than chance 的意思是 error less than half, 想一想也應該是 less. 這個錯誤浪費了老子不少時間.
[注 2]: 同理, 我將原文的 greater than half 改為了 less than half.

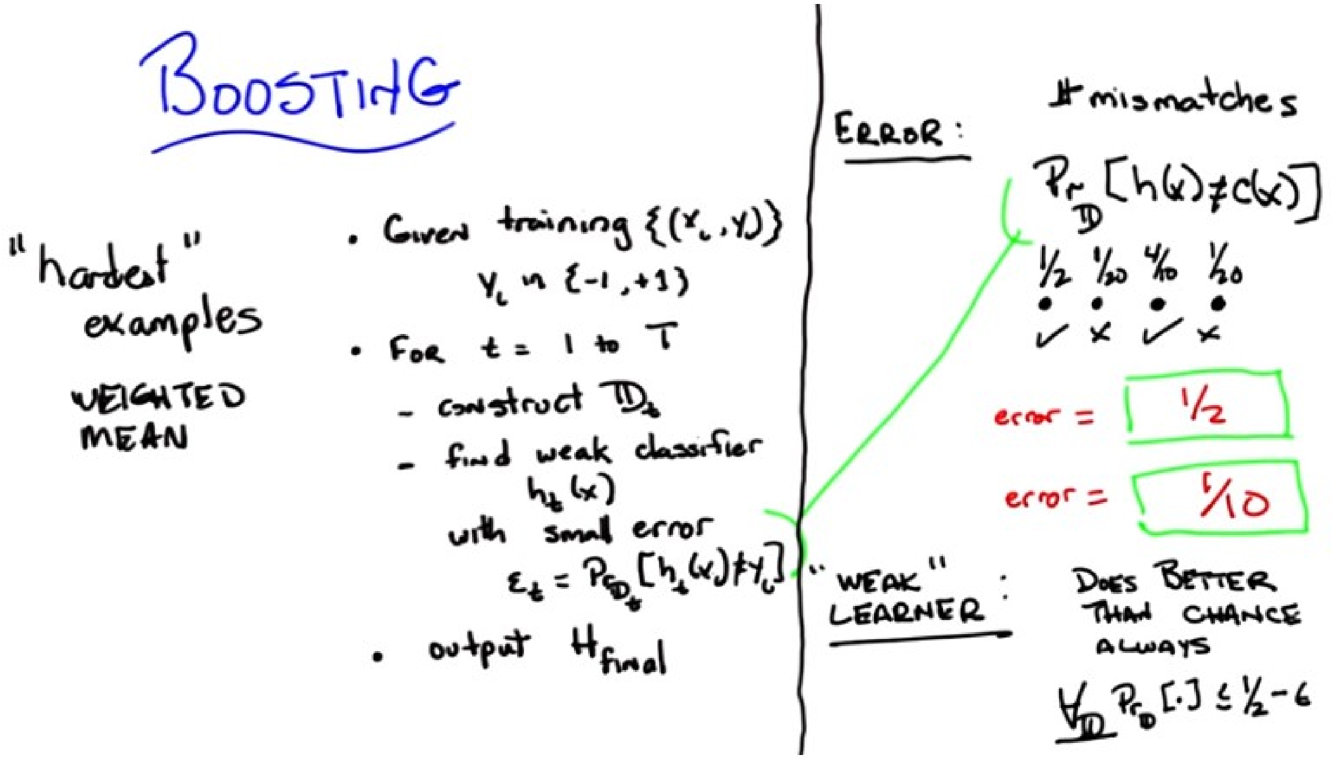
13. Okay Michael, you got answers for me?  >> Yeah, I think so. The first thing I notice is that if I put equal weights on all four examples, like I, I decided that instead of solving this problem by thinking, I would just try a couple examples, and see if I found things in both boxes. So, if I put equal weight on X1, X2, X3, X4.  >> Mm-hm.  >> Then hypothesis one, H1 gets three out of four correct, that's three quarters.  That's better than a half. So.  >> Well done.  >> Then I fill that in, in the good boxes, quarters all the way down.  >> That's a turtle, 'because it's turtles all the way down [LAUGH].  >> No, no, it's not though, it should be quarters all the way down. I thought you'd may be draw a quarter.  >> I, I can't draw a quarter, also I can't draw a turtle obviously but still.  >> [LAUGH] Agreed. Alright, good.  >> You'd think, anyone, do you think anyone listening to this is old enough to get turtles all the way down. >> Yeah, that's a great joke. Everybody knows that joke.  >> And if people don't know the joke, then we should pause this thing right now, and you should go look up turtles all the way down. And then come back. Okay.  >> It's a, it's a really great joke if you're computer scientist.  >> Yes, and if you don't think it's a good joke then you should probably be in a different field. Okay.  >> [LAUGH] >> What about the evil distribution?  >> So then I started to generate. Okay, well what if, the, the, the issue here is that, because we spread all the, the, the, the probability out in the first hypothesis really good. So I said okay, well let me put all the weight on the first example. The x1.  >> Okay. So what did that look like.  >> Now h1 did very badly. It gets, it's 100 percent error. And h2 is 100 percent error. But h3 is 0 percent errors. So so.  >> yes.  >> So, so putting it all putting all the weight on x1 is no good. And if you look x2, x3, x4, they all have the property that there's always a hypothesis that gets them right.  So I started to think, well maybe there isn't an evil distribution. And I kind of lucked into putting a half on both the first and the second one. because i figured that, that ought to work, but then i realized, oh wait

a second that's an evil distribution, because if you choose h1, h2, or h3, they all have exactly 50% error on the half a half distribution. >> Very good. So 1/2, 1/2, zero, zero, is a correct answer. >> Now I don't know if there's others. You know, certainly X, putting all the weight on X2 and X3 is no good, because H2 and H1 both get those. Putting all the weight on X3 and X4 are no good, because H1 gets all of those correct. In fact we have to have some weight on X1, right. Otherwise H1 is the way to go. >> Right. So, yeah. No, that's interesting. So what does that mean in this case? >> What do you mean, what does that mean? >> So what does this tell us about, how do we build a weak learner for this example (evilD 情況下)? >> So what it tells us is that since there is a distribution for which none of these hypotheses will be better than chance, there is no weak learner for this hypothesis space, on this instant set. >> Interesting. Now is there a way that we can, like, okay, so this example has no weak learner.



|  | h1 | h2 | h3 |
|---|---|---|---|
| x1 | ✗ | ✗ | ✓ |
| x2 | ✓ | ✓ | ✓ |
| x3 | ✓ | ✓ | ✗ |
| x4 | ✓ | ✗ | ✓ |

Is there a way to change this example so it would have a weak learner? >> Um...I'm sure there is. >> Like if we change that x2, x, h3, if that was a check instead of an X. >> Which one? x2 h3. >> So if we made that a green one, here I'll, I'll make it a green one. By using the power of computers. >> Woah, special effect. >> Yes. >> So now there's no way to put weight on any two things and have it fail. I don't know, my intuition now is that this, this should have a weak learner. Okay, well, how would we prove that? >> I don't know, but may be we should end this quiz. >> Yeah, I think, we should end this quiz. And leave it as an exercise to the listener. I'm pretty sure I can figure this out. By the way, we should point a couple of things here though, Michael. That one is that, the if it weren't the case, if we had more hypotheses and more examples. Perhaps an odd number of them and we have the x's and the y's in the right places then there'd be lots of ways to get weak learners for all the distributions just because you'd have more choices to choose from. What made this one particularly hard is that you only had three hypotheses and none of them was, not all of them were particularly good. >> Sure, yeah. I mean if you have a bun-, you can have many, many hypotheses and they're all pretty bad then you're not going to do very well. >> Well, it would depend upon if they're bad on very different things. But you're right, if you have a whole lot of hypotheses that are bad at everything, you're going to have a very hard time with a weak learner (可能是因為每個 hypothesis 都太瞥了, 所以無論論你用甚麼 distribution, 最後得到的正確率都小於 1/2, think of 數學期望). And if you have a whole bunch of hypotheses that are good at almost everything, then it's pretty easy to have a weak learner (由 22 段知, weak learner 其實是 does well 的人, 故本句話也是可以理解的). >> Interesting. Okay, this is more subtle than I thought. So that's, that's interesting. >> Right. So what the lesson you should take away from this is. If you were just, to think about it for 2 seconds you might think okay weak learner. That seems easy. And often it is, but if you think about for 4 seconds you realize that's actually a pretty strong condition. You're going to have to have a lot of hypotheses that are, many of which are going to have to do good on lots of different examples, or otherwise, you're not always going to be able to find one that does well no matter what the distribution is. So it's actually a fairly strong, and important condition.



Copy from 前面: a **weak learners** is, is a learner that no matter what the distribution is over your data, will do better than chance when it tries to learn labels on that data.

t 與其說表示某一時刻, 還不如說表示某一步, 或很多次中的某一次.
“find weak classifier h_t(x)”此句話驗證了 weak classifier 是一個 hypothesis.

14. All right Michael, so here's boosting in pseudo code. Okay, let me read it out to you then you can tell me if it makes sense. So we're given a training set. It's made up of a bunch of (x_i, y_i) pairs. You know, x is your input and y is your output. And for reasons that'll be clear in a moment all of your labels are either -1 or +1. Where minus one means not in class or plus one means you're in a class. So this is a binary classification task.

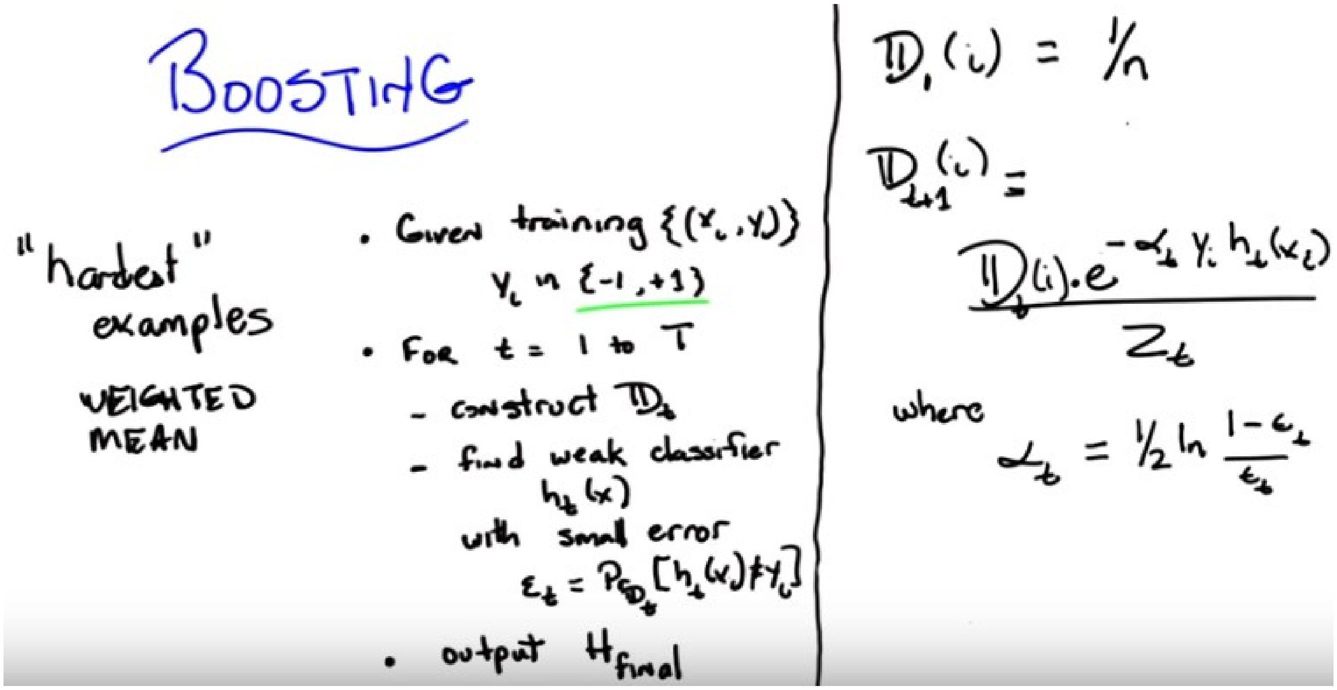
例如:
(x_1, y_1) = (A 這個東西, 是輛車)
(x_2, y_2) = (B 這個東西, 不是輛車)
(x_3, y_3) = (C 這個東西, 是輛車)
(x_4, y_4) = (D 這個東西, 不是輛車)
…...
注意所有的 y_i 都是真實(即正確)的結果

h_t(x)就是對結果的一個猜測, 即這麼一個列向量: (是輛車, 不是輛車, 不是輛車, 不是輛車...). 注意它對 C 這個東西猜錯了.

Dt 即這麼一個列向量: (10%, 20%, 15%, 30%, …), 意思是 A 這個東西出現在大街上的概率為 10%, ...

That make sense? >> So far. >> Okay. And then what you're going to do is, you're going to loop at every time step, let's call it lower-case t. From the first time step one, to some big time in the future. We'll just call it capital T and not worry about where it comes from right now. The first thing you're going to do is you're going to construct a distribution. And I'll tell you how in a minute, Michael. Okay, so, so, don't worry about it. And we'll just call that D_t. So, this is your distribution over your examples at some particular time t. Given that distribution, I want you to find a weak classifier. I want your weak learner to output some hypothesis. Let's call that h_t. The hypothesis that gets produced to that time step. And that hypothesis should have some small error. Let's call that error ε_t, because it's a small number. Such that it does well on the training set, given the particular distribution. So, I'm just rewriting my notion of error From, the other side of the screen. So at every time we want you to find a weak classifier [注]. That is, we want you to call some weak learner that returns some hypothesis. Lets call it h of t that has a small error. Lets call that epsilons of t. Which is to say ,that the probability of it being wrong that is disagreeing with the training label is small, with respect to the underlying distribution. >> So just to be clear there, the ε_t could be as big as slightly less than a half. Right? It doesn't have to be teeny, tiny. It could actually be, almost a half. But it can't be bigger than a half. >> That's right. And, and no matter what happens. Or even equal to a half. but, you know, we can assume, although it doesn't matter for the algorithm, that the learner is going to return the best one that it can, with some error. But regardless, it's going to have, it's going to satisfy the requirements of a weak learner, and all I've done is copy this notion of error over to here. Ok? >> Awesome! >> Ok. So, you're going to do that and you'll do that a whole bunch of times steps, constantly finding hypothesis at each time step [INAUDIBLE] with small error [INAUDIBLE] constantly making new. Distributions, and then eventually, you're going to output some final hypothesis. Which, I haven't told you yet how you're going to to get the final hypothesis. But that's the high level bit. Look at your training data, construct distributions, find a week classifier with low error. Keep doing that you have a bunch of them and then combine them somehow into some final hypothesis. And that's high level of algorithm for boosting, okay? >> Okay, but you've left out the two, two really important things, even the part from how you find we, weak classifier, which is where do we get this distribution and where do we get this, this final hypothesis. >> Right, so let me do that for you right now.

[注]: 原文為 So there are times we want you to find a weak classifier. 但我聽起來是 So at every time we want you to find a weak classifier.



BOOSTING

"hardest" examples

WEIGHTED MEAN

- Given training $\{(x_i, y_i)\}$
  $y_i$ in $\{-1, +1\}$
- For $t = 1$ to $T$
  - construct $D_t$
  - find weak classifier $h_t(x)$ with small error $\varepsilon_t = P_{D_t}[h_t(x_i) \neq y_i]$
- output $H_{final}$

$D_1(i) = 1/n$

$$D_{t+1}(i) = \frac{D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

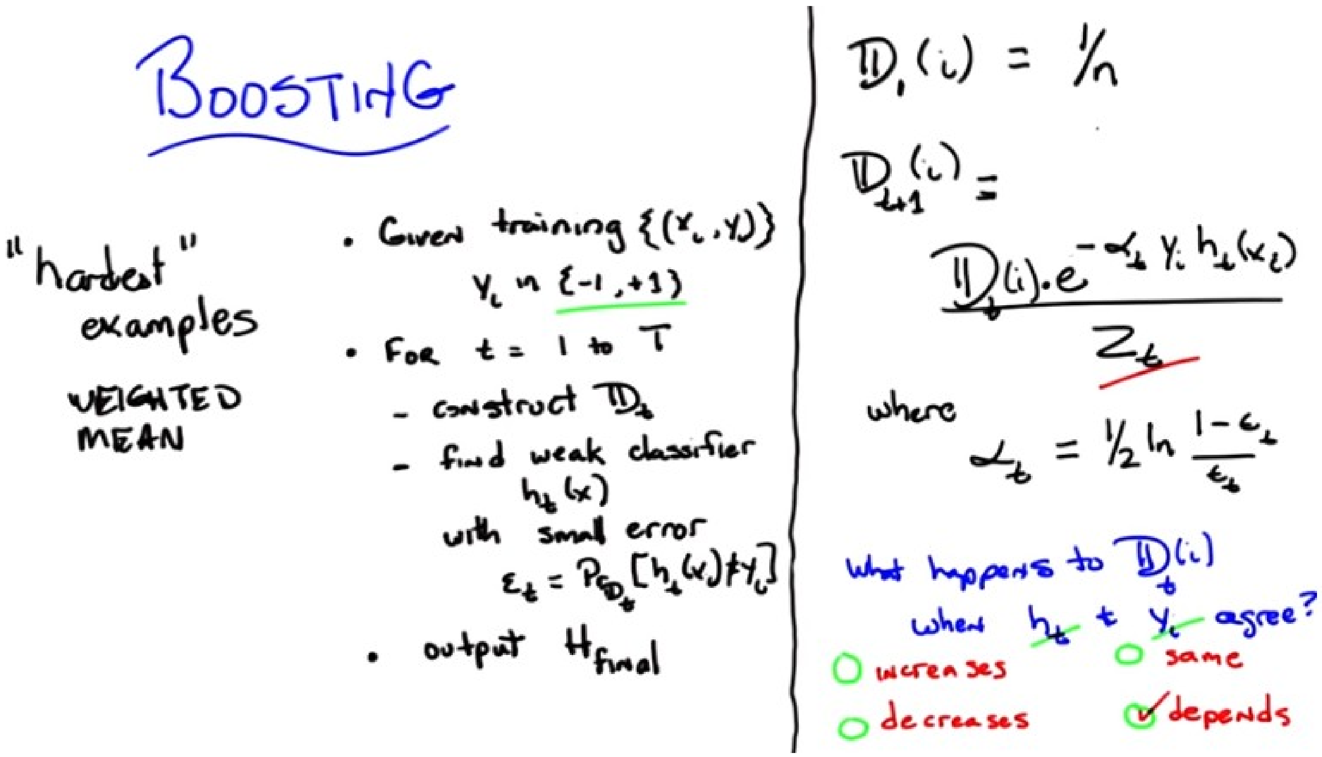
where $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

已驗證: 上圖右半邊的式子就是 Boosted Decision Tree 裡面的 AdaBoost.

15. Okay Michael, you've called my bluff. You, you said I've left out the most important parts, and you are right. So, I'm going to tell you ,how to construct the most important parts. Let's start, with the distribution. So, let's start with the base case, and that is the distribution, at the beginning of time, D_1 (i). So, this distribution, is going to be over each of the examples and those examples, are ,indexed, so, over i. And I'm simply going to set that distribution, to be uniform. So, how many examples do we have (即(x_i, y_i) pair 的數量), Michael? Let's pick, let's pick a letter. Let's call it n. >> Okay. >> Why not, cause we do that for everything else and I'm going to say that for every single one of the examples they happen one over int times, that is uniform distribution. Now, why do I do that, because I have no reason to believe, for the purposes of this algorithm, that any given example is better than any other example, more important than any other example, harder than other example or anything else. I know nothing. So, see if you can learn over all of the examples. You with me? >> Yeah, cause I feel like if it actually solves that problem, we're just done. So [CROSSTALK] >> [CROSSTALK] Yes and, and that's what you always want. But that's the easy case. So I start out with uniform distribution, that's what you usually do ,when you don't know anything. But, what are you going to do ,while your in the middle? So, here's what I am going to do Michael. At every time step t, I'm going to construct, a new distribution, D_t+1 (i). Okay so, here's how we're going to construct the distribution at every time step. Okay? I'm going to create the new distribution, T plus 1 to be E for each example, I. - to be the old distribution, and times T, times E to the minus alpha T, times Y suvai, times H of sub-T, of X of I, all divided by Z sub T. [LAUGH] So that's pretty obvious right? [LAUGH] So what do each of those terms mean? I mean ,I know it's intuitively obvious ,even to the most casual observer, but ,let me just try to explain what each of the parts mean. So, we know that the D is our distribution and it's some number, where, over all the examples, it adds up to one. And it's a stand-in, we know, because I said this at the beginning, for how important a particular ,example is, how often we expect to see it. And that's the trick that we're using with distributions. So, I'm going to take the old distribution for an example, of, for a particular example. And I'm going to either make it bigger or smaller, based upon, how well, the current hypothesis, does, on that particular example. So, there's a cute little trick here, we know that h of t always returns, a value ,of either minus one or plus one, because ,that's how we define

our training set, you always have a label of minus one or plus one. So, ht is going to return minus one or plus one for a particular x of i. Y of i which is the label with respect to that, example ,is also always going to be plus one or minus one. And alpha t is a constant, which I will get into a little bit later just right now think of it as a number. And so what happens ,Michael, if the hypothesis ,at time t for a particular example x of i agrees ,with the label ,that is associated with that x of i? >> Well, hang on, you say the alpha's a number. Is it a positive number? A between 0 and 1 number? A negative number? What kind of number? Does, does it not matter. I think it matters. >> That's a good question. It, it matters eventually. But right now, that number is always, positive. >> Positive, alright. So, like, a [UNKNOWN]. Almost like a learning rating kind of thing, maybe. >> It's a learning rating kind of thing, sort of. >> Alright, so, good. So the y, okay, I see, I see. So, the y times the h is going to be. 1 if they agree, and minus 1 if they disagree. >> Exactly, so if they both say positive 1, positive 1 times positive 1 is 1. If they both say negative 1, negative 1 times negative 1 is 1. So, it's exactly what you say when they agree, that number is 1. And when they disagree, that number is minus 1. α_t, which I define below, is always a positive number (因為 ε_t 總是小於 1/2, 這就是為何要找 weak learner!). You can trust me on this. The error is always between 0 and 1. And it just turns out that the natural log of 1 minus a number between 0 and 1 over that number, always gives you a positive number. And if you don't believe me, you should play around with the numbers till you convince yourself. So, that's going to be some positive number. So, that means when they agree, that whole product will be positive. Which means ,you'll be raising e to minus some number when they disagree that product will be negative which means you'll be raising e to some positive number. So, let's imagine they agree. So you're going to be re raising, e ,to minus some number, what's going to happen to the relative weight of d sub t of i?

16. So, Michael wants us to do a quiz. Because Michael likes quizzes cause he thinks you like quizzes. And so, I want you to answer this question before Michael gets a chance to. So just to be clear here's the question again. What happens to the distribution over a particular example i when the hypothesis ht that was output by the example. Agrees with the particular label, Y-sub-i. Okay, so we have four possibilities when they agree. One is the probability of you seeing that particular example increases. That is, you increase the value of D-sub-t on i. Or the probability of you seeing that example decreases. That is, the number d of t of i goes down, or it stays the same when they agree or well it depends on exactly what's going on with the old value of d and alpha and all these other things. So ,you can't really give just one of those other three answers. So those are your possibilites. The other radio buttons [LAUGH] only one of them is right. And go.



BOOSTING

"hardest" examples

WEIGHTED MEAN

- Given training $\{(x_i, y_i)\}$ $y_i$ in $\{-1,+1\}$
- For $t = 1$ to $T$
  - construct $D_t$
  - find weak classifier $h_t(x)$ with small error $\varepsilon_t = P_{D_t}[h_t(x_i) \neq y_i]$
- output $H_{final}$

$$D_1(i) = 1/n$$

$$D_{t+1}(i) = \frac{D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

where $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

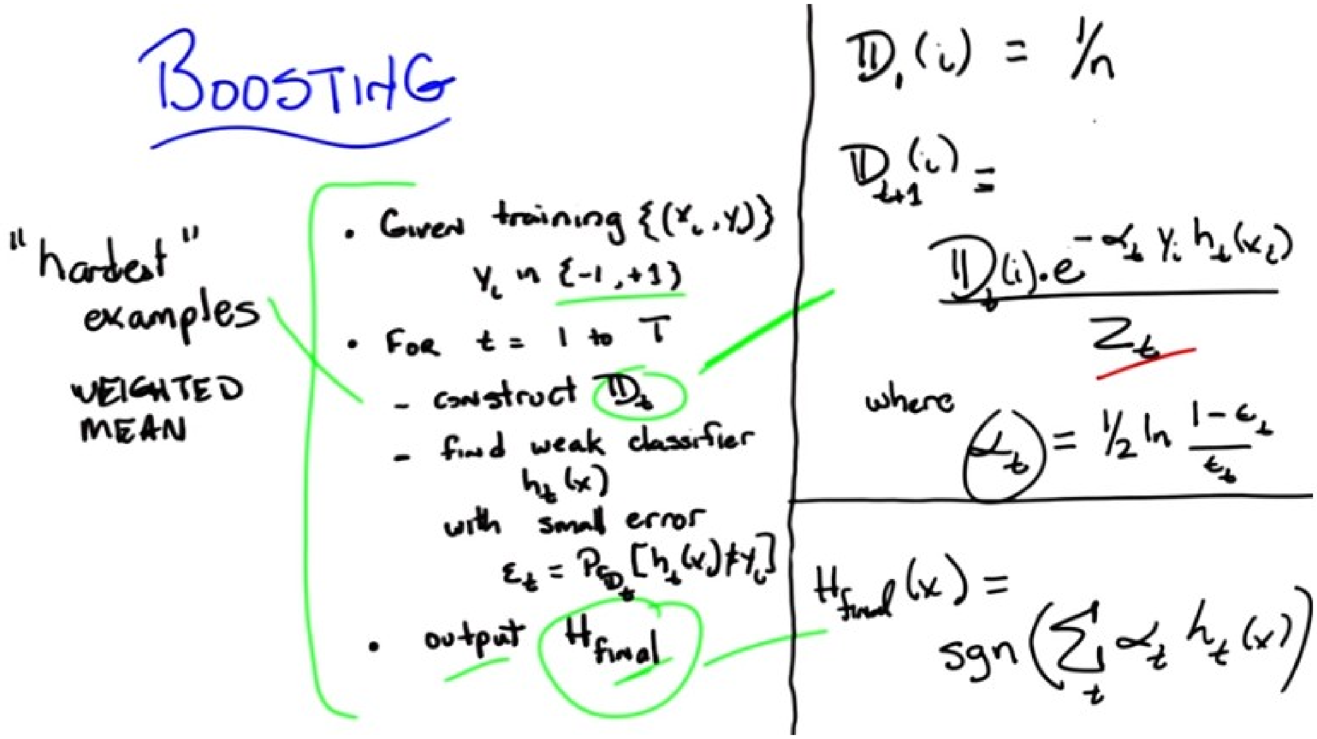
What happens to $D_t(i)$ when $h_t$ + $y_i$ agree?

○ increases ○ same

○ decreases [✓ depends]

17. Okay Michael what's the answer. >> Alright, so you kind of were walking us through it, but basically if Yi and Ht agree, that means they're both negative or they're both positive. They're equal to each other. So when we multiply them together we get one. One times whatever our alpha thing is, some positive number is going to be positive. We're negating that, so it's negative E to the negative something is something between zero and one. Less than, less than one. So, that's going to scale it down. So, it looks like. And you know assuming that everything else goes ok with, with the way that ,uh, the normalization happens right? It seems like it could be depends on the normalization. >> So by the way. That's a good point. The the Z_t, is in fact, what ever normalization constant you need at time t, in order to make it all work out to be a distribution. >> Correct. Then not going to, not going to change. >> True. >> But is some of them are correct and some of them incorrect, the ones that are correct are going to decrease. And the ones that are incorrect are going to increase. >> That's right, so what's the answer to the quiz. >> Depends. >> That's true, it does. That's exactly the right answer. It depends, on what else is going on, you're correct. Now. >> But I feel like it should be decreases, like that's really, mainly what happens. >> That's also fair. The answer is, if this one is correct, that is they agree, and you disagree on at least some of them, at least one, one other example, it will in fact decrease. So I could ask a similar question, which is, well what happens when they disagree and at least one other example agrees? Then what happens? >> Yeah, then they, then that should increase. Oh. >> Right. >> Oh. It's going to put more weight on the ones that it's getting wrong. >> Exactly. And the ones that it's getting wrong must be the ones that are harder. Or at least that's the underlying idea. All right, Michael, so you got it? So you understand what the equation's for? >> Yeah, it look. It seemed really scary at first but it seems you know marginally less scary now because all that it's doing, it's doing it in a particular way. I don't know why it's doing it in this particular way. But all it seems to be doing is. The answers that it, it had, it was getting wrong... It puts more weight on those and the ones that its getting right, it puts less weight on those and then you know, the loop goes around again and it tries to make a new classifier. >> Right, and since the ones that its getting wrong are getting more and more weight but we are guaranteed or at least we've assumed that we have a weak learner that will always do better than chance. On ,ah, any distribution, it means that you'll always be able to

output some learner that can get some of the ones 「that you were getting wrong」 right.


BOOSTING
"hardest" examples
WEIGHTED MEAN
• Given training {(x_i, y_i)}
y_i in {-1, +1}
• For t = 1 to T
- construct D_t
- find weak classifier h_t(x)
with small error
ε_t = Pr_{D_t}[h_t(x_i) ≠ y_i]
• output H_final
D_1(i) = 1/n
D_{t+1}(i) =
D_t(i)·e^{-α_t y_i h_t(x_i)} / Z_t
where α_t = ½ ln (1-ε_t)/ε_t
H_final(x) = sgn(Σ_t α_t h_t(x))

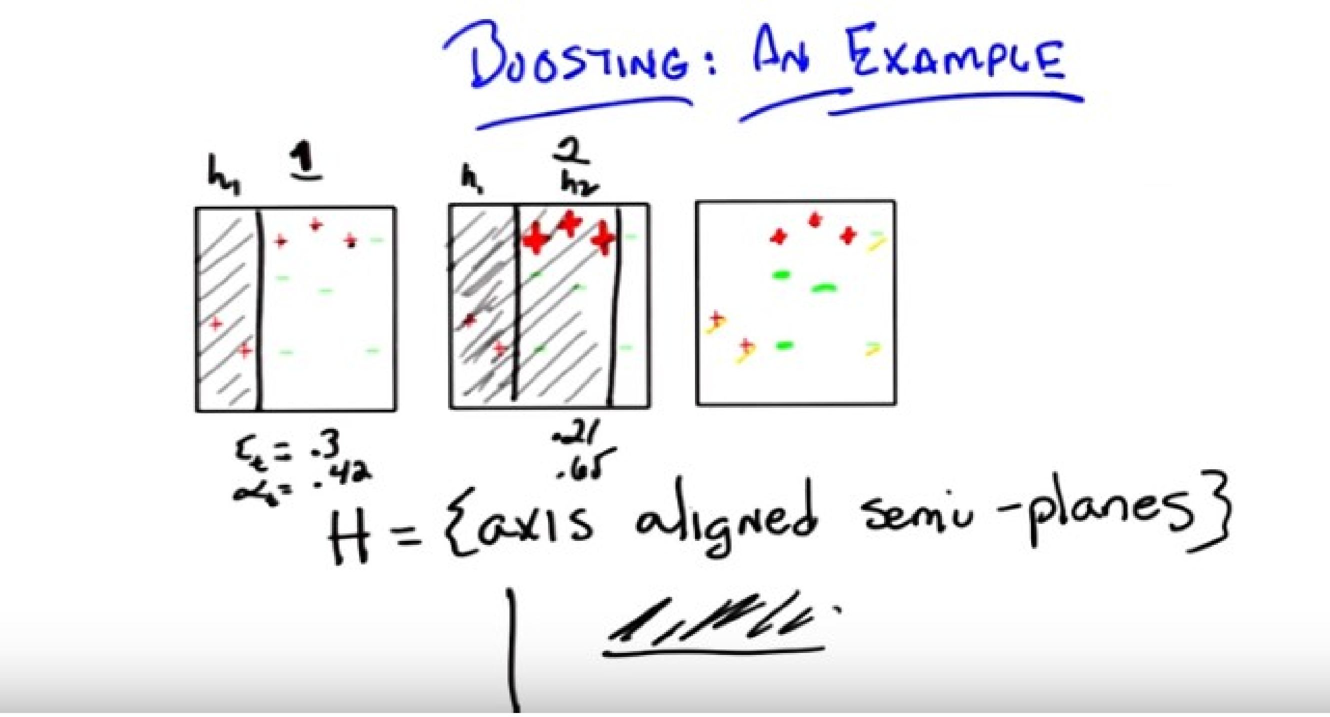
注意 H_final(x)是 x 的函數，H_final(x)表達式中有 h_t(x). 但由上面幾行式子知 h_t 是 x_i 的函數: h_t(x_i). 故 H_final(x)表達式中的 x 應該就是 x_i, 只是忘了寫下標 i. 即: H_final 是 x_i 的函數: H_final(x_i)表示由算法得到的 x_i 的最終 output 是多少, H_final(x_i)只能取-1 或+1.

注意 H_final 表達式中的 h_t 就是 weak learner.

若 boosting 用到 decision tree 中，則就是 Boosted Decision Tree. 某 t 對應的一個 h_t(x_i)即「BDT 中 forest 裡面的一個 tree 給出的結果」. 注意 BDT 中最後所有的 tree vote, 即上圖中由 h_t(x_i)加權平均得出 H_final(x_i).

18. So that ties together this, what constructed D does for you, and connecting it to the hardest examples. So now, that gets us to a nice little trick where we can talk about how we actually output our final example. So, the way you construct your final example, they way you do that combination in the step is basically by doing a weighted average. And the weight Is going to be based upon this alpha sub T. So the final hypothesis is just the sgn function of the weighted sum of all of the rules of thumb, all of the weak classifiers that you've been picking up over all of these time steps. Where they're weighted by the alpha sub T's. And remember, the α_t is one half of the natural log of one minus epsilon T over epsilon T. That is to say, it's a measure of how well you're doing with respect to underlining error. So, you get more weight if you do well Then if you do less well or you get less weight. So what does this look like to you? Well its a weighted average based on how well you're doing or how well each of the individual hypotheses are doing and then you pass it through (sgn:) a thresh holding function where if its below zero you say you know what? Negative. And if its above zero you say you know what? Positive. And if its zero you just throw up your hands and And return zero. In other words, you return literally the sign of the number. So you are throwing away information there, and I'm not going to tell you what it is now, but when we go to the next lesson it;s going to turn out that that little bit of information you throw away is actually pretty important. But that's just a little bit of a teaser. We'll get

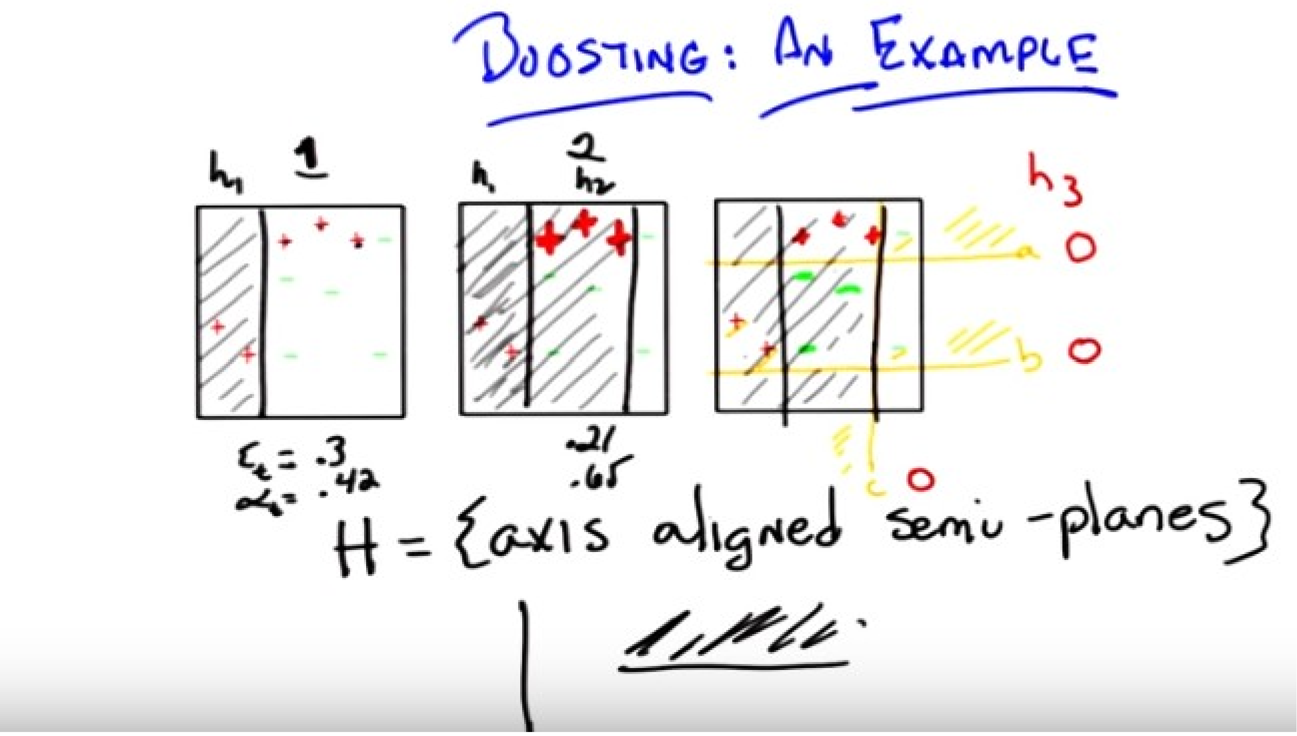
back to that there. Okay so, this is boosting, Michael. There's really nothing else to it. You have a very simple algorithm, which can be written down in a couple of lines. The hardest parts are constructing the distribution, which I show you how to do over here, and then simply bringing everything together, which I show you how to do over here.  >> Alright yeah, I think it doesn't seem so bad and I feel like I could code this up, but I would be a little happier if I had a handle on what the, why alpha is the way that it is.  >> Well there's two answers. The first answer is. You use natural logs because you're using exponentials and that's always a cute thing to do. And of course, you're using the error term as a way of measuring how good the hypothesis is. And the second answer is, it's in the reading you were supposed to have done. [LAUGH] So, go back and read the paper now that you've listened to this and you will have a much better understanding of what it's trying to tell you.  >> Thanks >> You're welcome. I'm about helping others Michael you know that.



19. So, Michael, I want to try to convince you other than the fact that it's an algorithm with symbols that, it sort of works, at, at least informally. And then, I'm going to do what I always do and refer you to actually read the, the, the text to get the, the details.  But before I do that, I wanted to go through an example if you think that would help.  >> I would like an example.  >> Okay. So, let's go through an example. So, here's a very simple example. So, I got three little boxes on the screen. Can you see them? >> Yeah.  >> Now, they're the same boxes. I've drawn them up here beforehand because I'm going to solve this problem in only three steps.  >> Hey those boxes are really nice, did you get help from our trusty course developer?  >> I did in fact did get help from our trusty course developer. And when I say help, I mean he did this.  >> Oh thanks Push Car.  >> Yes Push Car is wonderful. Now what's really cool about this is that Push Car is already let you know that we're going to be able to do this in 3 simple steps. And I'm going to be able to animate it. Or at least hopefully it'll look animated by the time, [LAUGH] we're done with all the editing. So just pay attention to the first box for now, you have a bunch of data points; red pluses and green minuses, which is the opposite of what we usually do Push Car. But either way it's red pluses and green minuses.  [LAUGH] With the appropriate labels and they all live in this, this part of the plane. By the way, what do you call a part of the plane? I know you have line segments, what's like, a sub part of a plane?  >> Looks like a square to me.  >> Yes it is, but I

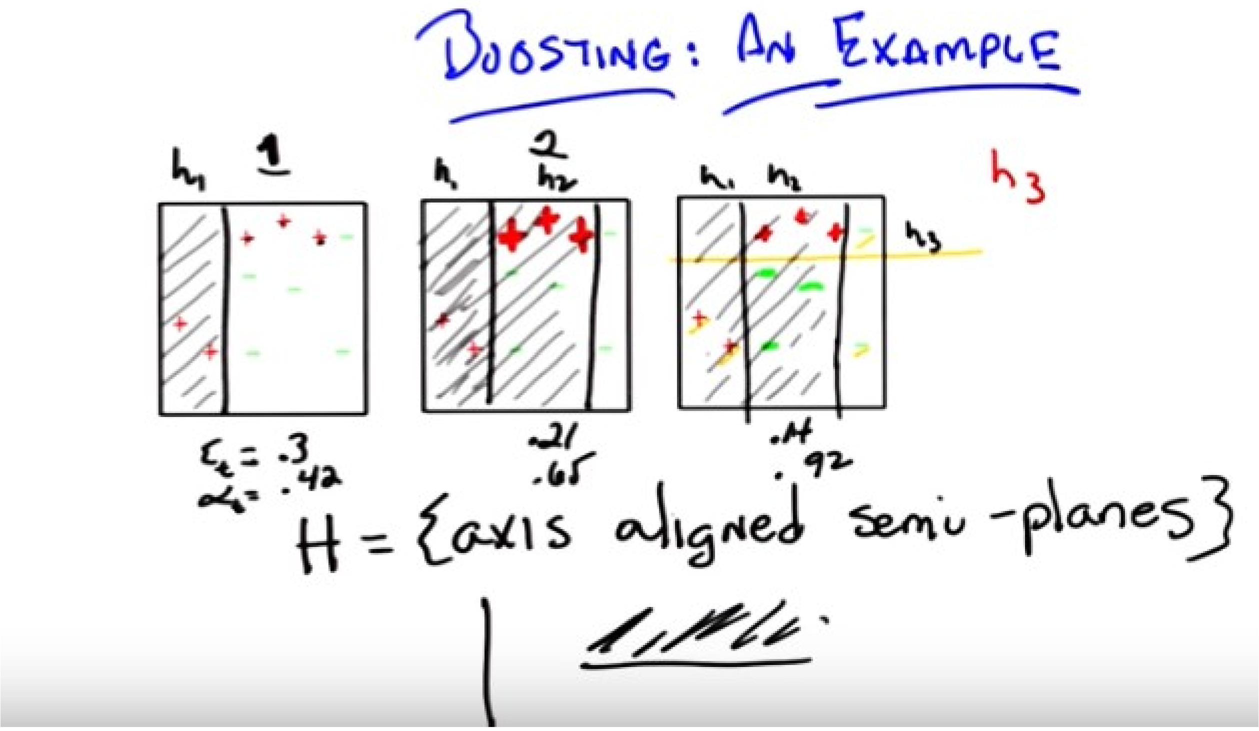
mean, what do you call them? You, you don't call it a plane segment, do you? What do you call it? >> A region. >> A square region, fine. So it's a square region on a plane. And we want to figure out how to be able to correctly classify these examples. Okay, so that is nothing new there. We just want to be able to come up with something. So now we have to do what we did like in the quiz is that we have to specify what our hypothesis space is. So here's our hypothesis space. So the hypothesis space is the set of axis aligned semi-planes. You know what that means? >> Mm, no. >> Well for the purpose of this example it means, I'm going to draw a line, either horizontal or vertical and say that everything on one side of that line is positive and everything on the other side of that line is negative (這可以輕易地寫為一個 hypothesis 的形式: {++--+-+-}, 即若某點在 line 的這一邊 則為+, 若在另一邊 則為-). >> I see. Okay, good. >> Right. And they're axes align because it's only horizontal and vertical, and they're semi-planes because the positive part of it is only in part of the plane. Okay, so I'm going to just walk through what boosting would end up doing with this particular example or what a boosting might do with this particular example given that you have a learner that always chooses between axis aligned semi planes. Okay? >> Yeah. >> So let's imagine we ran our boosting algorithm now in the beginning it's step 1 all of the examples look the same because we have no particular reason to say any are more important than the other, any are easier or harder than the other. And that's just the algorithm we had before We run through and we ask our learner to return some hypotheses that does well in classifying the examples. It turns out that though there are many, and in fact there are an infinite number of possible hypotheses you could return. One that works really well is one that looks like a vertical line that separates the first two data points from the rest. >> That is what I was guessing. >> Of course it was. And what I'm saying here is that everythign to the left of this line is going to be positive and everything to the right is going to be negative. So if you look at this what does this hypothesis do? So it gets correct, correctly labeled positive. The two pluses to the left. Right? >> Correct. >> And it gets correct all of the minuses as well. >> Correct. >> Right? But it gets wrong the three pluses on the right side. So it gets, this wrong, this wrong, and this wrong. >> Right, the Three Plusketeers. >> Exactly. [LAUGH] The Three Plusketeers. That's actually pretty good. So I'm just you know I'm just going to ask you to trust me here but it turns out that the specific error here is 0.3 (因為有 3 個點錯了, 總共 10 個點) and if you stick that into our little alpha you end up, our little, our little formula for alpha, you end up with alpha equal to 4.2. >> That's not obvious to me but. >> Is is See, see, see it's not always obvious. >> [LAUGH] >> Okay. Good. So there you go and that's just what happens when you stick this particular set in there. So now we're going to construct the next distribution. Right? And what's going to happen in the next distribution? >> So the one's that it got right should get less weight and the one's that it got wrong should get more weight so those three plusketeers should become more prominent somehow. >> That's exactly what happens. They become, I'm just going to draw them as much thicker and bigger to kind of emphasise that they're getting bigger, and it's going to turn out that everything else is going to get smaller which is a lot harder to draw here. So i'm just going to kind of leave them their size, so they sort of get normalized away. Okay? >> I would guess as to what the next plane should be. I think that we should cut it. Underneath those pluses but above the green minuses. And that should get us three errors. The two pluses on the left and the minus on the top will be wrong. But they have less weight than the three pluses we got right, so this going to be better than the previous one. >> So, that's possibly true. But it's not what the learner output. >> Oh! >> Let me tell you what the learner did output though. This learner output by putting a line to the right of the three pluses, because he's gotta get those right in saying that everything to the left is in fact, positive. So, does that seem like a reasonable one to you? >> Well, it does better than half. I guess that's really all what we're trying to do, but it does seem to do worse than what I suggested. >> Well, let's see, it gets the three that mattered that you were really, really doing poorly right but then so did yours. And it only, and it picks up still the other two which it was getting right. And it gets wrong these three minus' which aren't worth so much. So is that worse than what you suggested? No, it gets wrong, oh, the three minuses. Oh, it gets correct those two red pluses on the left. So it gets three things wrong. So that's just as good as what I suggested. Okay, I agree. >> Okay good. So the area of this step by the way, turns out to be 0.21 and the alpha at this [INAUDIBLE] step turns out to be 0.65. So that's pretty interesting, so we got a bunch of these right and a bunch of these wrong. So what's going to happen to the distribution over these examples. >> Alright, the ones that it, again, the ones that it got wrong should get pushed up in weight and which ones are those, those are the, the three green minuses in the middle patch >> Right. >> They should become more prominent. The pluses, the three, the three plusketeers should become less prominent than they were but it still might be more prominent than they were in the beginning. And maybe because in fact the alpha, let's see the alpha is bigger so, it will have actually a bigger effect on bringing it down. >> Yeah I guess so, but it, there'll still be more prominent than the other ones that haven't been bumped. >> Yeah the ones that you, the, the two, the two red pluses on the left have, you've never gotten them wrong. >> Hm. >> So they're really going to disappear. So, if we do, If I do my best to, If I do my best to kind of draw that you're still, you're going to have. These pluses are going to be a little bit bigger than the other pluses, but they're going to be smaller than they were before. The two, the three greens in the middle are going to be bigger than they were before. But those two pluses are going to be even smaller, and these two minuses are going to be smaller. So, what do you think the third hypothesis should be. >> Quiz. >> Oh, I like that.

20. Okay, so Michael wanted to have a quiz here, 'because Michael again, likes those sort of things and, and I like to please Michael. So, we came up with three possibilities, one of which we hope is right. >> [LAUGH] >> And I've labeled them here in orange, A, B, and C and put little radio boxes next to 'em, so you could select 'em. So which of those three hypotheses are, is a good one to pick next? So, A is a horizontal line that says everything above it should be a plus. B is a, another horizontal line that says everything above it should be a plus. And C is a vertical line, like the last two hypotheses that we found, that says everything to the left should be a plus. So, which do you think is the right one? Go.

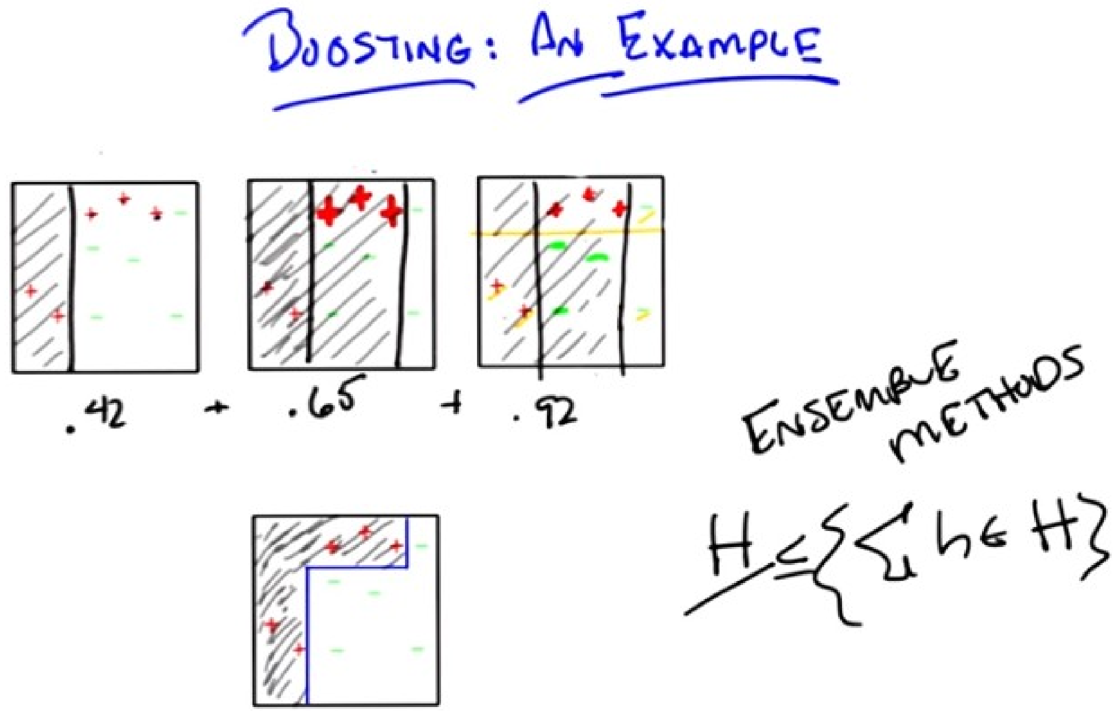


21. >> Alright Michael, what's the answer? >> Alright so of those others, well C is pretty good,

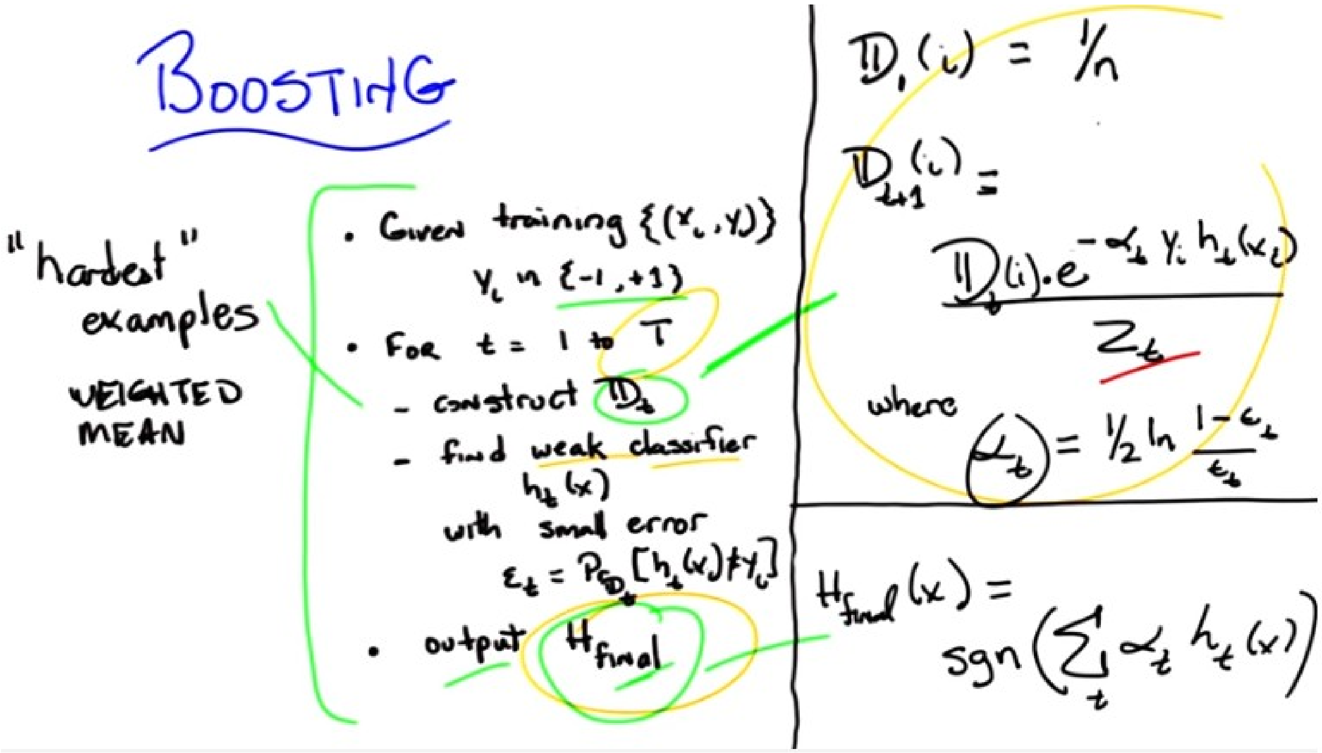
because it does separate the pluses from the minuses. We, we even liked it so much we used it in round two.  >> Mh-hm.  >> But it doesn't as good to me as A, because A actually does a good job of separating the very, the more heavily weighted points. So I would, I would say A.  >> So in fact that is what our little learning system shows.  It shows A.



Now, through the trick of animation, I leave you with A. And that is exactly the right answer. By the way, Michael, if you look at these three hypothesis and their weights, you end up with something kind of interesting. So if you look at this third hypothesis that's chosen here, turns out they have a very low error, you'll notice that the errors are going down over time, by the way, of 0.14. And it has a much higher alpha of 0.92.



Now if you look at these weights and you add them up (由前面知, 這就是找 H_final 的方法), you end up with a cute little combination. So, let me draw that for you. Okay Michael, so I cleaned up a little bit so that you could see it. If you take each of the three hypothesis that we produced, and you weight them accordingly, you end up with the bottom figure. >> No way. >> Absolutely. >> That's. Kind of awesome. So what you're saying is that, even though we were only using half planes, or, or axis-aligned semi planes, for all the weak learners, that at the end of the day it actually kind of bent the line around and captured the positive and negative examples perfectly. >> Right. Does that remind you of anything else we've talked about in the past? >> Sh. Everything. Nothing. No, I dunno, I mean so with, with decision trees you can make the shapes like that, and >> That's true. >> And the fact that we're doing a weighted combination of things reminds me of the neural net. >> Yeah. And it should remind you of one other thing. >> I'm imagining that you want me to say nearest neighbors, but I can't quite make the connection. >> Well, you recall in our discussion with nearest neighbors, when we did weighted nearest neighbor. In particular we did weighted linear regression, we were able to take a simple hypothesis, add it together in order to get a more complicated hypothesis. >> That's true, because it's local. >> Right, exactly because it's local, and this is a general feature of Ensemble methods that if you try to look at just some particular hypothesis class. Let's just call it H, because you're doing weighted averages over hypotheses drawn from that hypothesis class. This hypothesis class is almost all low, is at least as complicated as this hypothesis class and often is more complicated. So you're able to be more expressive, even though you're using simple hypotheses, because you're combining them in some way. >> I'm not surprised that you can combine simple things to get complicated things. But I am surprised that you can combine them just with sums. And get complicated things because sums often act very, you know, sort of, friendly. Right it's a linear combination not a nonlinear combination. >> Actually, Michael part of the reason you get something nonlinear here is because you're passing it through a non-linearity at the end. >> The sine. >> Yea, that's a good thing, we should, we should ponder that.



BOOSTING

"hardest" examples

WEIGHTED MEAN

- Given training $\{(x_i, y_i)\}$
  $y_i$ in $\{-1, +1\}$
- For $t = 1$ to $T$
  - construct $\mathbb{D}_t$
  - find weak classifier $h_t(x)$
    with small error
    $\varepsilon_t = \Pr_{\mathbb{D}_t}[h_t(x_i) \neq y_i]$
- output $H_{final}$

$$\mathbb{D}_1(i) = 1/n$$

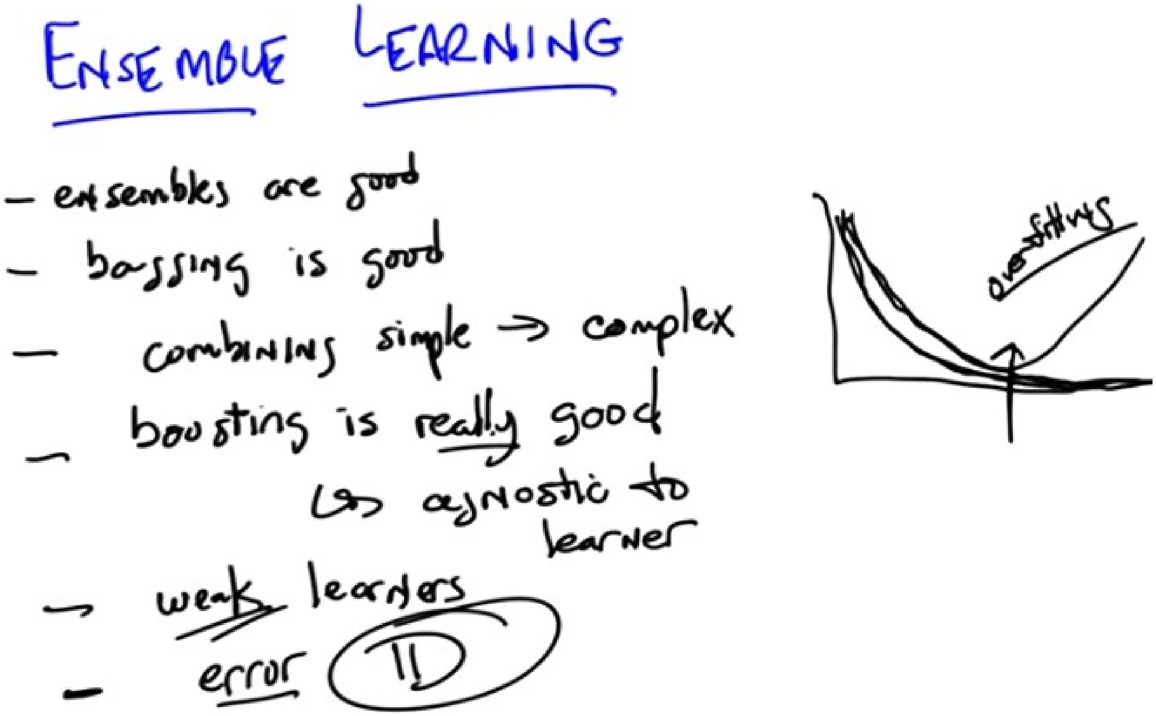
$$\mathbb{D}_{t+1}(i) = \frac{\mathbb{D}_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

where $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

$$H_{final}(x) = \operatorname{sgn}\left(\sum_t \alpha_t h_t(x)\right)$$

22. >> Okay Michael, so we've done our little example. I want to ask you a quick question and try to talk something through with you and then we can start to wrap up. Okay. >> Awesome. >> Alright, so, here is my quick question. Now, in the reading, which I know you've read, there's a proof. That shows that boosting not only, you know, does pretty things with axis of line semi-planes, but also that it will converge to good answers and that it will find good combined hypotheses. You know, we could go look at the reading and write down a proof that shows that boosting does well. Umm. And there's one in the reading. Or we could talk about an intuition. So if someone were to come up to you. If a student were to find you somewhere and said, I read the proof, I'm kind of getting it, but do you have a good sort of intuition about why boosting pins will do well? What do you think you would tell them? Could you think of something simple? I've been struggling with this for a while. >> No. [LAUGH]. >> Okay, well, then let me try something on you and you can tell me if it sort of makes sense. So this is just an intuition for why, for why boosting pins will do well. Okay, so what does boosting do? Okay. Boosting basically says, if I have some examples that I haven't been able to classify well, I'm going to re-rate all my examples. So that the ones I don't do well become increasingly important. Right, that's what boosting does. Yes? >> Yes. >> Right, that's what this whole, whole bit of D is all about. It's all about re-weighting based on difficulty and hardest. And we know that we have the notion of a weak learner. That no matter what happens for whatever distribution, we're always going to be able to find some hypothesis that does well (為何 weak learner 是 does well? 因為它的 error 小於 1/2 啊, 而且由公式可看出, 只要小於 1/2 就可以了, 注意 error 越小就越 does well). So, if I'm trying to understand why boosting in the end, why the final hypothesis that I get at the end, is going to do well. I can try to get a feeling for that by asking, well, under what circumstances would it not do well? So, if it doesn't do well, then that means there has to be a bunch of examples that it's getting wrong, right? >> Mm hm. >> That's what it would mean not to do well, agreed? >> Yeah. >> Okay. So how many things could it not get right? How many things could it misclassify? How many things could it get incorrect? Well, I'm going to argue Michael, that, that number has to be small. There cannot be a lot of examples that it gets wrong. So do you want to know why? Do you want to know my reasoning for why? >> Yeah. >> So, here's

my reasoning, let's imagine I had a number of examples at the end of this whole process. I've done it T times. I've gone through this many times and I have some number of examples that I'm getting wrong. If I were getting those examples wrong, then I was getting them wrong in the last time step, right? And, since I have a distribution and I re-normalize, and it has to be the case that at least half of the time, more than half of the time I am correct, that number of things I'm getting wrong has to be getting smaller over time. Because let's imagine that was at a stage where I had a whole bunch of them wrong. Well, then I would naturally renormalize them with a distribution so that all of those things are important. But if they were all important, the ones that I was getting wrong, the next time I run a learner, I am going to have to get at least half of them right, more than half of them are right (因為 weak learner). Is that make sense? >> It does, but it, but what scares me is, okay, why can't it just be the case that the previous ones which were getting right start to get more wrong as we shift our energy towards the errors. >> Yeah, why is that? >> I don't know. But did you wanna,are we, we working up to some kind of you know, log any kind of thing where each time you are knocking off half of them and therefore. >> I don't know. Do you remember the proof. >> The proof. >> I mean what goes on is that you get, sort of, this exponentially aggressive weighting over examples, right? >> Yeah. >> And you're driving down the number of things you get wrong. Sort of exponentially quickly, over time. That's why boosting works so well and works so fast. >> I get that we're, the we're quickly ramping up the weights on the hard ones. I don't get why that's causing us to get fewer things wrong over time. So like, when you should, in your, in your example that you worked through, that had the error in the alphas and the errors kept going down and the alphas kept going up. >> Right. >> Like, is that necessarily the case? >> Well, what would be the circumstances under which it isn't the case? How would you ever go back and forth between examples? Well, certainly it's the case that if you keep getting something, right, you will, get them. Well, so here's what happens over time, right. Is that over time, every new hypothesis, it gets to get a vote, based upon how well it does on the last, difficult let's say, distribution. So the ones that are even if the ones that you were getting right you start to get wrong, they are going to, if you get them increasingly wrong, that error's going to go down and you're going to get less of a vote, right. Because e sub T is over the current distribution. And it's not over the sum of the voted, over all the examples you've ever seen. >> Understand. >> So does that make sense? Is that right? >> I don't know. I don't have the intuition, it seems like it could be, you know, could we keep shifting the distribution. It could be that the error is going up. Like if the air could be low, why can't we just make it low from the beginning. >> Right. >> Like, I feel like the arrow should be going up, because we're, we're asking it harder and harder questions as we go. >> No, no, no, because we're asking it harder and harder questions, but even though we're asking it harder and harder questions, it's forced to be able to do well on those hard questions. It's forced to, because it's a weak learner. I mean that's why having, being able to always, that's why having a weak learner is such a powerful thing. >> But why couldn't we like on, on iteration 17, have something where the weak learner works right at the edge of it's abilities, and it just comes back with something that's a half minus epsilon. >> That's fine. But it has to always be able to do that. If it's a half minus epsilon, the things it's getting wrong will have to go back down again. >> No, no I understand that. What I'm saying is that, why would the error go down each iteration. >> Well, it doesn't have to, but it shouldn't be getting bigger. >> Why shouldn't it be getting bigger? >> So, imagine, imagine, imagine the case that you're getting, right. You, you are working at the edge of your abilities. You get half of them right. Roughly and half of them wrong, the ones you got wrong would become more important, so the next time round you're going to get those right, versus the other ones. So you could cycle back and forth I suppose, in the worst case, but then you're just going to be sitting around, always having a little bit more information. So your error will not get worse, you'll just have different ones that are able to vote on that do well on different parts of the space. Right? Because you're always forced to do better than chance. So. >> Yeah but that, that's not the same as saying that we're forced to get better and better each iteration. >> That's right, it's not. >> So it's, yeah again, I don't see that, that property just falling out. >> Well, I don't see it falling out either, but then I haven't read the proof in like seven, eight, nine years. >> Well, I feel like it should be, it should be something like, look we had, look at what the, so okay, so we generate a new distribution, what is the previous, what's the previous classified error on this distribution, it better be the case. I mean if it were the case that we always return the best classifier that I could imagine trying to use that but. >> Well we, well we don't, we don't require that. >> Yeah, I mean, it's just finding one that's epsilon minus, or a half minus epsilon. >> Right, so let's, let's see if we can take the simple case, we got three examples, right, and you're bouncing back and forth and you want to construct something so that you always do well on two of them. And then poorly on one, kind of a thing, and that you keep bouncing back and forth. So let's imagine that you have one-third, one-third, one-third, and your first thing gets the first two right and the last one wrong. So you have an error of a third. And you make that last one more likely and the other two less likely. Suitably normalized, right? >> Yep. >> So now, your next one, you want to somehow bounce back and have it decide that it can miss, so lets say you missed the third one. So you, you get the third one right. You get the second one right but you get the first one wrong. What's going to happen? Well, three is going to go down. You're still going to, well you won't have a third error actually. You'll have less than a third error because you had to get one of the ones you were getting right wrong, you had to get the one you were getting wrong right. So your error is going to be at least an example I just gave. Less than a third. So, if your error is less and a third, then the weighting goes up more. And so, the one that you just got wrong goes up, doesn't go back to where it was before. It becomes even more important than it was when you had a uniform distribution. So the next time around, you have to get that one right, but it's not enough to break a half. So you're going to have to get something else right as well, and the one in the middle that you were getting right isn't enough. So you'll have to get number three right as well. >> Interesting. >> Right? And so, it's really hard to cycle back and forth between different examples, because you're exponentially weighting how important they are. Which means, you're always going to have to pick up something along the way. Because the ones that you, coicidentally, got right two times in a row. Become so unimportant. It doesn't help you to get those right. Whereas, the ones that you've gotten wrong, in the past. You've got to, on these cycles. Pick up some of them in order to get you over a half. >> Mmm >> And so, it is very difficult for you to cycle back and forth. >> Interesting. >> And that kind of makes sense, right? If you think about it in kind of an information gain sense, because what's going on there is you're, you're basically saying you must pick up information all the time. >> Hm. And then your non uni. Well uniform is the wrong word but you are kind of. You know, non-linearly using that information in some way. So that kind of works. It makes some sense to me, but I think that in the end what has to happen is you. You, there must be just a few examples in a kind of weighted sense that you're getting wrong. And so if I'm right, that as you, as you move through each of these cycles, you're weighting in such a way that you have to be picking up things you've gotten wrong in the past. So in other words, it's not enough to say, only the things that are hard in the last set, are the ones that I have to do better. You must also be picking up some of the things that you've gotten wrong earlier more than you were getting right. Because there's just not enough information in the one's that you're getting right all the time, because by the time you get that far along, the weight on them is near zero and they don't matter. >> Interesting. >> And then if you say, well, Charles, I could cycle back by always getting those wrong, yes, but then if you're getting those wrong, they're going to pull up and you're going to have to start getting those right too. And so, over time, you've gotta not just pick out things that do better than a half. But things that do well on a lot of the data. Because there's no way for all of the possible distributions for you to do better than chance otherwise. >> Cool.



ENSEMBLE LEARNING

- ensembles are good
- bagging is good
- combining simple → complex
- boosting is really good
  ↳ agnostic to learner
- weak learners
- error TD

overfitting

23. Okay, Michael, so that was a great conversation, what have we learned? >> Alright, well we talked about ensemble learning which was the idea of instead of just learning one thing, if it's good to learn once, it's even better to learn multiple times, >> In multiple ways. >> The simple version that we concentrated on first was this notion of bagging. Where what we did is instead of just learning on the whole data set, we would sub-sample bunch of examples from the training set, different ways, and train up different classifiers or different learners on each of those and then merge them together with the average. >> Okay, so if I can summarize that, we learned that ensembles are good. [LAUGH] >> We learned that even simple ensembles like bagging are good. >> We talked about the fact that by using this kind of ensemble approach, you can take simple learners or simple classifiers and merge them together and get more complicated classifiers. >> Mm, yeah, so we can take. We can. Combining simple gives you complex. Anything else? >> And we talked about the idea of boosting where you can Oh, maybe this is why it's called boosting. You can take something that has possibly very high error but always less than a half, and turn it into something that has very low error. >> So we learned that boosting is really good. And, we talked a little bit about why, that's good. By the way, there's a whole bunch of other details here too, right? Boosting also has the advan, as does bagging Not only has these little properties you've talked about before, but it tends to be very fast. It's agnostic to the learner. As you noticed, that in no time, did we say, try to take advantage of what the actual learner was doing. Just that it was, in fact, a weak learner. >> Hm. >> So I think that's important. It's agnostic. >> Meaning you can plug in any learner you want? >> Yeah. So long as it's a weak learner. So there's something we learned about. We learned about weak learners that we defined with that meant. And, we also talked about ,um, what error really, really means. With respect to some kind of underlying distribution. What do you think Michael? >> That seems like useful stuff. >> These are useful stuff to me. I'm going to throw one more thing at you, Michael, before I let you go. Okay, you ready? >> Yep. >> Here's a simple fact. About boosting that turns out in practice. You know our favorite little over-fitting example. Do you know how over-fitting works? You have a training line that tends to get better, and better, and better. Maybe even going down to zero error. But then you have test error Which gets better and better

and at some point it starts to get worse. >> Mm. >> And at that point you have over fitting and I think, Michael, you asserted it at some point or maybe I asserted that ,you always have to worry about over fitting. Over fitting is just the kind of fact of life. You got to come up with ways to deal with it or sort of over believing your data. Well, what if I told you that in practice When you run boosting, even as you run it over time so that your training error keeps getting better and better and better and better, it also turns out that your testing error keeps getting better and better and better and better and better and better and better. >> That seems too good to be true. >> It does seem too good to be true. It turns out it's not too good to be true. And I have an explanation for it. >> Tell me. >> Not until next time. >> alright, see you then. >> See you then. Bye.