MachineLearning-Lecture09

Instructor (Andrew Ng): All right, good morning. Just one quick announcement, first things for all of your project proposals, I've read through all of them and they all look fine. There is one or two that I was trying to email back comments on that had slightly questionable aspects, but if you don't hear by me from today you can safely assume that your project proposal is fine and you should just go ahead and start working on your proposals. You should just go ahead and start working on your project. Okay, there's many exciting proposals sent in on Friday and so I think the proposal session at the end of the quarter will be an exciting event.

Okay. So welcome back. What I want to do today is start a new chapter in between now and then. In particular, I want to talk about learning theory. So in the previous, I guess eight lectures so far, you've learned about a lot of learning algorithms, and yes, you now I hope understand a little about some of the best and most powerful tools of machine learning in the [inaudible]. And all of you are now sort of well qualified to go into industry and though powerful learning algorithms apply, really the most powerful learning algorithms we know to all sorts of problems, and in fact, I hope you start to do that on your projects right away as well.

You might remember, I think it was in the very first lecture, that I made an analogy to if you're trying to learn to be a carpenter, so if you imagine you're going to carpentry school to learn to be a carpenter, then only a small part of what you need to do is to acquire a set of tools. If you learn to be a carpenter you don't walk in and pick up a tool box and [inaudible], so when you need to cut a piece of wood do you use a rip saw, or a jig saw, or a keyhole saw whatever, is this really mastering the tools there's also an essential part of becoming a good carpenter. And what I want to do in the next few lectures is actually give you a sense of the mastery of the machine learning tools all of you have. Okay?

And so in particular, in the next few lectures what I want to is to talk more deeply about the properties of different machine learning algorithms so that you can get a sense of when it's most appropriate to use each one. And it turns out that one of the most common scenarios in machine learning is someday you'll be doing research or [inaudible] a company. And you'll apply one of the learning algorithms you learned about, you may apply logistic regression, or support vector machines, or Naïve Bayes or something, and for whatever bizarre reason, it won't work as well as you were hoping, or it won't quite do what you were hoping it to.

To me what really separates the people from – what really separates the people that really understand and really get machine learning, compared to people that maybe read the textbook and so they'll work through the math, will be what you do next. Will be in your decisions of when you apply a support vector machine and it doesn't quite do what you wanted, do you really understand enough about support vector machines to know what to do next and how to modify the algorithm? And to me that's often what really separates the great people in machine learning versus the people that like read the text book and so they'll [inaudible] the math, and so they'll have just understood that. Okay? So what I want to do today – today's lecture will mainly be on learning theory and we'll start to talk about some of the theoretical results of machine learning. The next lecture, later this week, will be on algorithms for sort of [inaudible], or fixing some of the problems that the learning theory will point out to us and help us understand. And then two lectures from now, that lecture will be almost entirely focused on the practical advice for how to apply learning algorithms. Okay? So you have any questions about this before I start? Okay.

So the very first thing we're gonna talk about is something that you've probably already seen on the first homework, and something that alluded to previously, which is the bias variance trade-off. So take ordinary least squares, the first learning algorithm we learned about, if you [inaudible] a straight line through these datas, this is not a very good model. Right. And if this happens, we say it has underfit the data, or we say that this is a learning algorithm with a very high bias, because it is failing to fit the evident quadratic structure in the data. And for the prefaces of [inaudible] you can formally think of the bias of the learning algorithm as representing the fact that even if you had an infinite amount of training data, even if you had tons of training data, this algorithm would still fail to fit the quadratic function – the quadratic structure in the data. And so we think of this as a learning algorithm with high bias. Then there's the opposite problem, so that's the same dataset. If you fit a fourth of the polynomials into this dataset, then you have – you'll be able to interpolate the five data points exactly, but clearly, this is also not a great model to the structure that you and I probably see in the data.

And we say that this algorithm has a problem – excuse me, is overfitting the data, or alternatively that this algorithm has high variance. Okay? And the intuition behind overfitting a high variance is that the algorithm is fitting serious patterns in the data, or is fitting idiosyncratic properties of this specific dataset, be it the dataset of housing prices or whatever. And quite often, they'll be some happy medium of fitting a quadratic function that maybe won't interpolate your data points perfectly, but also captures multi-structure in your data than a simple model which under fits. I say that you can sort of have the exactly the same picture of classification problems as well, so lets say this is my training set, right, of positive and negative examples, and so you can fit logistic regression with a very high order polynomial [inaudible], or [inaudible] of X equals the sigmoid function of – whatever. Sigmoid function applied to a tenth of the polynomial. And you do that, maybe you get a decision boundary like this. Right. That does indeed perfectly separate the positive and negative classes, this is another example of how overfitting, and in contrast you fit logistic regression into this model with just the linear features, with none of the quadratic features, then maybe you get a decision boundary like that, which can also underfit. Okay.

So what I want to do now is understand this problem of overfitting versus underfitting, of high bias versus high variance, more explicitly, I will do that by posing a more formal model of machine learning and so trying to prove when these two twin problems – when each of these two problems come up. And as I'm modeling the example for our initial foray into learning theory, I want to talk about learning classification, in which H of X is equal to G of data transpose X. Okay? So the learning classifier. And for this class I'm going to use, Z – excuse me – I'm gonna use G as indicator Z grading with zero. With apologies in advance for changing the notation yet again, for the support vector machine lectures we use Y equals minus one or plus one. For learning theory lectures, turns out it'll be a bit cleaner if I switch back to Y equals zero-one again, so I'm gonna switch back to my original notation. And so you think of this model as a model forum as logistic regressions, say, and think of this as being similar to logistic regression, except that now we're going to force the logistic regression algorithm, to opt for labels that are either zero or one. Okay? So you can think of this as a classifier to opt for labels zero or one involved in the probabilities. And so as usual, let's say we're given a training set of M examples. That's just my notation for writing a set of M examples ranging from I equals one through M. And I'm going to assume that the training example is XIYI. I've drawn IID, from sum distribution, scripts D. Okay? [Inaudible]. Identically and definitively distributed and if you have – you have running a classification problem on houses, like features of the house comma, whether the house will be sold in the next six months, then this is just the priority distribution over features of houses and whether or not they'll be sold. Okay?

So I'm gonna assume that training examples we've drawn IID from some probability distributions, scripts D. Well, same thing for spam, if you're trying to build a spam classifier then this would be the distribution of what emails look like comma, whether they are spam or not. And in particular, to understand or simplify – to understand the phenomena of bias invariance, I'm actually going to use a simplified model of machine learning. And in particular, logistic regression fits this parameters the model like this for maximizing the law of likelihood. But in order to understand learning algorithms more deeply, I'm just going to assume a simplified model of machine learning, let me just write that down. So I'm going to define training error as – so this is a training error of a hypothesis X subscript data. Write this epsilon hat of subscript data. If I want to make the dependence on a training set explicit, I'll write this with a subscript S there where S is a training set. And I'll define this as, let's see. Okay. I hope the notation is clear. This is a sum of indicator functions for whether your hypothesis correctly classifies the Y – the IFE example.

And so when you divide by M, this is just in your training set what's the fraction of training examples your hypothesis classifies so defined as a training error. And training error is also called risk. The simplified model of machine learning I'm gonna talk about is called empirical risk minimization. And in particular, I'm going to assume that the way my learning algorithm works is it will choose parameters data, that minimize my training error. Okay? And it will be this learning algorithm that we'll prove properties about. And it turns out that you can think of this as the most basic learning algorithm, the algorithm that minimizes your training error.

It turns out that logistic regression and support vector machines can be formally viewed as approximation cities, so it turns out that if you actually want to do this, this is a nonconvex optimization problem. This is actually – it actually [inaudible] hard to solve this optimization problem. And logistic regression and support vector machines can both be viewed as approximations to this nonconvex optimization problem by finding the convex approximation to it. Think of this as similar to what algorithms like logistic regression are doing. So let me take that definition of empirical risk minimization and actually just rewrite it in a different equivalent way. For the results I want to prove today, it turns out that it will be useful to think of our learning algorithm as not choosing a set of parameters, but as choosing a function. So let me say what I mean by that. Let me define the hypothesis class, script h, as the class of all hypotheses of – in other words as the class of all linear classifiers, that your learning algorithm is choosing from. Okay? So H subscript data is a specific linear classifier, so H subscript data, in each of these functions – each of these is a function mapping from the input domain X is the class zero-one. Each of these is a function, and as you vary the parameter's data, you get different functions. And so let me define the hypothesis class script H to be the class of all functions that say logistic regression can choose from. Okay.

So this is the class of all linear classifiers and so I'm going to define, or maybe redefine empirical risk minimization as instead of writing this choosing a set of parameters, I want to think of it as choosing a function into hypothesis class of script H that minimizes – that minimizes my training error. Okay? So – actually can you raise your hand if it makes sense to you why this is equivalent to the previous formulation? Okay, cool. Thanks. So for development of the use of think of algorithms as choosing from function from the class instead, because in a more general case this set, script H, can be some other class of functions. Maybe is a class of all functions represented by viewer network, or the class of all – some other class of functions the learning algorithm wants to choose from. And this definition for empirical risk minimization will still apply. Okay? So what we'd like to do is understand whether empirical risk minimization is a reasonable algorithm. Alex?

Student:[Inaudible] a function that's defined by G of data TX, or is it now more general?

Instructor (Andrew Ng): I see, right, so lets see – I guess this – the question is H data still defined by G of phase transpose X, is this more general? So –

Student:[Inaudible]

Instructor (Andrew Ng): Oh, yeah so very – two answers to that. One is, this framework is general, so for the purpose of this lecture it may be useful to you to keep in mind a model of the example of when H subscript data is the class of all linear classifiers such as those used by like a visectron algorithm or logistic regression. This – everything on this board, however, is actually more general. H can be any set of functions, mapping from the INFA domain to the center of class label zero and one, and then you can perform empirical risk minimization over any hypothesis class.

For the purpose of today's lecture, I am going to restrict myself to talking about binary classification, but it turns out everything I say generalizes to regression in other problem as well. Does that answer your question?

Student: Yes.

Instructor (Andrew Ng): Cool. All right. So I wanna understand if empirical risk minimization is a reasonable algorithm. In particular, what are the things we can prove about it? So clearly we don't actually care about training error, we don't really care about making accurate predictions on the training set, or at a least that's not the ultimate goal. The ultimate goal is how well it makes – generalization – how well it makes predictions on examples that we haven't seen before. How well it predicts prices or sale or no sale outcomes of houses you haven't seen before.

So what we really care about is generalization error, which I write as epsilon of H. And this is defined as the probability that if I sample a new example, X comma Y, from that distribution scripts D, my hypothesis mislabels that example. And in terms of notational convention, usually if I use – if I place a hat on top of something, it usually means – not always – but it usually means that it is an attempt to estimate something about the hat. So for example, epsilon hat here – this is something that we're trying – think of epsilon hat training error as an attempt to approximate generalization error. Okay, so the notation convention is usually the things with the hats on top are things we're using to estimate other quantities. And H hat is a hypothesis output by learning algorithm to try to estimate what the functions from H to Y – X to Y. So let's actually prove some things about when empirical risk minimization will do well in a sense of giving us low generalization error, which is what we really care about. In order to prove our first learning theory result, I'm going to have to state two lemmas, the first is the union vowel, which is the following, let A1 through AK be K event. And when I say events, I mean events in a sense of a probabilistic event that either happens or not. And these are not necessarily independent.

So there's some current distribution over the events A one through AK, and maybe they're independent maybe not, no assumption on that. Then the probability of A one or A two or dot, dot, dot, up to AK, this union symbol, this hat, this just means this sort of just set notation for probability just means "or." So the probability of at least one of these events occurring, of A one or A two, or up to AK, this is S equal to the probability of A one plus probability of A two plus dot, dot, dot, plus probability of AK. Okay? So the intuition behind this is just that – I'm not sure if you've seen Venn diagrams depictions of probability before, if you haven't, what I'm about to do may be a little cryptic, so just ignore that. Just ignore what I'm about to do if you haven't seen it before. But if you have seen it before then this is really – this is really great – the probability of A one, union A two, union A three, is less than the P of A one, plus P of A two, plus P of A three. Right. So that the total mass in the union of these three things [inaudible] to the sum of the masses in the three individual sets, it's not very surprising.

It turns out that depending on how you define your axioms of probability, this is actually one of the axioms that probably varies, so I won't actually try to prove this. This is usually written as an axiom. So sigmas of avitivity are probably measured as this – what is sometimes called as well. But in learning theory it's commonly called the union balance – I just call it that. The other lemma I need is called the Hufting inequality. And again, I won't actually prove this, I'll just state it, which is – let's let Z1 up to ZM, BM, IID, there may be random variables with mean Phi. So the probability of ZI equals 1 is equal to Phi. So let's say you observe M IID for newly random variables and you want to estimate their mean. So let me define Phi hat, and this is again that notation, no convention, Phi hat means – does not attempt – is an estimate or something else. So when we define Phi hat to be 1 over M, semper my equals one through MZI. Okay? So this is our attempt to estimate the mean of these Benuve random variables by sort of taking its average. And let any gamma be fixed.

Then, the Hufting inequality is that the probability your estimate of Phi is more than gamma away from the true value of Phi, that this is bounded by two E to the next of two gamma squared. Okay? So just in pictures – so this theorem holds – this lemma, the Hufting inequality, this is just a statement of fact, this just holds true. But let me now draw a cartoon to describe some of the intuition behind this, I guess. So lets say [inaudible] this is a real number line from zero to one. And so Phi is the mean of your Benuve random variables. You will remember from – you know, whatever – some undergraduate probability or statistics class, the central limit theorem that says that when you average all the things together, you tend to get a Gaussian distribution. And so when you toss M coins with bias Phi, we observe these M Benuve random variables, and we average them, then the probability distribution of Phi hat will roughly be a Gaussian lets say. Okay? It turns out if you haven't seen this up before, this is actually that the cumulative distribution function of Phi hat will converse with that of the Gaussian. Technically Phi hat can only take on a discreet set of values because these are factions one over Ms. It doesn't really have an entity but just as a cartoon think of it as a converse roughly to a Gaussian.

So what the Hufting inequality says is that if you pick a value of gamma, let me put S one interval gamma there's another interval gamma. Then the saying that the probability mass of the details, in other words the probability that my value of Phi hat is more than a gamma away from the true value, that the total mass – that the total probability mass in these tails is at most two E to the negative two gamma squared M. Okay? That's what the Hufting inequality – so if you can't read that this just says – this is just the right hand side of the bound, two E to negative two gamma squared. So balance the probability that you make a mistake in estimating the mean of a Benuve random variable.

And the cool thing about this bound – the interesting thing behind this bound is that the [inaudible] exponentially in M, so it says that for a fixed value of gamma, as you increase the size of your training set, as you toss a coin more and more, then the worth of this Gaussian will shrink. The worth of this Gaussian will actually shrink like one over root to M. And that will cause the probability mass left in the tails to decrease exponentially, quickly, as a function of that. And this will be important later. Yeah?

Student:

Does this come from the central limit theorem [inaudible].

Instructor (Andrew Ng): No it doesn't. So this is proved by a different – this is proved – no – so the central limit theorem – there may be a version of the central limit theorem, but the versions I'm familiar with tend – are sort of asymptotic, but this works for any finer value of M. Oh, and for your – this bound holds even if M is equal to two, or M is [inaudible], if M is very small, the central limit theorem approximation is not gonna hold, but this theorem holds regardless. Okay? I'm drawing this just as a cartoon to help explain the intuition, but this theorem just holds true, without reference to central limit theorem.

All right. So lets start to understand empirical risk minimization, and what I want to do is begin with studying empirical risk minimization for a [inaudible] case that's a logistic regression, and in particular I want to start with studying the case of finite hypothesis classes. So let's say script H is a class of K hypotheses. Right. So this is K functions with no – each of these is just a function mapping from inputs to outputs, there's no parameters in this. And so what the empirical risk minimization would do is it would take the training set and it'll then look at each of these K functions, and it'll pick whichever of these functions has the lowest training error. Okay?

So now that the logistic regression uses an infinitely large – a continuous infinitely large class of hypotheses, script H, but to prove the first row I actually want to just describe our first learning theorem is all for the case of when you have a finite hypothesis class, and then we'll later generalize that into the hypothesis classes. So empirical risk minimization takes the hypothesis of the lowest training error, and what I'd like to do is prove a bound on the generalization error of H hat. All right. So in other words I'm gonna prove that somehow minimizing training error allows me to do well on generalization error.

And here's the strategy, I'm going to – the first step in this prove I'm going to show that training error is a good approximation to generalization error, and then I'm going to show that this implies a bound on the generalization error of the hypothesis of [inaudible] empirical risk minimization. And I just realized, this class I guess is also maybe slightly notation heavy class round, instead of just introducing a reasonably large set of new symbols, so if again, in the course of today's lecture, you're looking at some symbol and you don't quite remember what it is, please raise your hand and ask. [Inaudible] what's that, what was that, was that a generalization error or was it something else? So raise your hand and ask if you don't understand what the notation I was defining.

Okay. So let me introduce this in two steps. And the empirical risk strategy is I'm gonna show training errors that give approximation generalization error, and this will imply that minimizing training error will also do pretty well in terms of minimizing generalization error. And this will give us a bound on the generalization error of the hypothesis output by empirical risk minimization. Okay? So here's the idea. So lets even not consider all the hypotheses at once, lets pick any hypothesis, HJ in the class script H, and so until further notice lets just consider there one fixed hypothesis. So pick any one hypothesis and let's talk about that one.

Let me define ZI to be indicator function for whether this hypothesis misclassifies the IFE example – excuse me – or Z subscript I. Okay? So ZI would be zero or one depending on whether this one hypothesis which is the only one I'm gonna even consider now, whether this hypothesis was classified as an example. And so my training set is drawn randomly from sum distribution scripts d, and depending on what training examples I’ve got, these ZIs would be either zero or one. So let’s figure out what the probability distribution ZI is. Well, so ZI takes on the value of either zero or one, so clearly is a Benuve random variable, it can only take on these values.

Well, what’s the probability that ZI is equal to one? In other words, what’s the probability that from a fixed hypothesis HJ, when I sample my training set IID from distribution D, what is the chance that my hypothesis will misclassify it? Well, by definition, that’s just a generalization error of my hypothesis HJ. So ZI is a Benuve random variable with mean given by the generalization error of this hypothesis. Raise your hand if that made sense. Oh, cool. Great.

And moreover, all the ZIs have the same probability of being one, and all my training examples I’ve drawn are IID, and so the ZIs are also independent – and therefore the ZIs themselves are IID random variables. Okay? Because my training examples were drawn independently of each other, by assumption.

If you read this as the definition of training error, the training error of my hypothesis HJ, that’s just that. That’s just the average of my ZIs, which was – well I previously defined it like this. Okay? And so epsilon hat of HJ is exactly the average of MIID, Benuve random variables, drawn from Benuve distribution with mean given by the generalization error, so this is – well this is the average of MIID Benuve random variables, each of which has meaning given by the generalization error of HJ.

And therefore, by the Hufting inequality we have to add the probability that the difference between training and generalization error, the probability that this is greater than gamma is less than to two, E to the negative two, gamma squared M. Okay? Exactly by the Hufting inequality.

And what this proves is that, for my fixed hypothesis HJ, my training error, epsilon hat will with high probability, assuming M is large, if M is large than this thing on the right hand side will be small, because this is two Es and a negative two gamma squared M. So this says that if my training set is large enough, then the probability my training error is far from generalization error, meaning that it is more than gamma, will be small, will be bounded by this thing on the right hand side. Okay?

Now, here’s the [inaudible] tricky part, what we’ve done is approve this bound for one fixed hypothesis, for HJ. What I want to prove is that training error will be a good estimate for generalization error, not just for this one hypothesis HJ, but actually for all K hypotheses in my hypothesis class script H. So let’s do it – well, better do it on a new board. So in order to show that, let me define a random event, let me define AJ to be the event that – let me define AJ to be the event that – you know, the difference between training and generalization error is more than gamma on a hypothesis HJ. And so what we put on the previous board was that the probability of AJ is less equal to two E to the negative two, gamma squared M, and this is pretty small. Now, What I want to bound is the probability that there exists some hypothesis in my class script H, such that I make a large error in my estimate of generalization error. Okay? Such that this holds true. So this is really just that the probability that there exists a hypothesis for which this holds. This is really the probability that A one or A two, or up to AK holds. The chance there exists a hypothesis is just well the priority that – for hypothesis one and make a large error in estimating the generalization error, or for hypothesis two and make a large error in estimating generalization error, and so on. And so by the union bound, this is less than equal to that, which is therefore less than equal to – is equal to that. Okay?

So let me just take one minus both sides – of the equation on the previous board – let me take one minus both sides, so the probability that there does not exist for hypothesis such that, that. The probability that there does not exist a hypothesis on which I make a large error in this estimate while this is equal to the probability that for all hypotheses, I make a small error, or at most gamma, in my estimate of generalization error. In taking one minus on the right hand side I get two KE to the negative two gamma squared M. Okay? And so – and the sign of the inequality flipped because I took one minus both sides. The minus sign flips the sign of the equality. So what we're shown is that with probability – which abbreviates to WP – with probability one minus two KE to the negative two gamma squared M. We have that, epsilon hat of H will be – will then gamma of epsilon of H, simultaneously for all hypotheses in our class script H.

And so just to give this result a name, this is called – this is one instance of what's called a uniform conversions result, and the term uniform conversions – this sort of alludes to the fact that this shows that as M becomes large, then these epsilon hats will all simultaneously converge to epsilon of H. That training error will become very close to generalization error simultaneously for all hypotheses H. That's what the term uniform refers to, is the fact that this converges for all hypotheses H and not just for one hypothesis. And so what we're shown is one example of a uniform conversions result. Okay? So let me clean a couple more boards. I'll come back and ask what questions you have about this. We should take another look at this and make sure it all makes sense. Yeah, okay. What questions do you have about this?

Student:

How the is the value of gamma computed [inaudible]?

Instructor (Andrew Ng): Right. Yeah. So let's see, the question is how is the value of gamma computed? So for these purposes – for the purposes of this, gamma is a constant. Imagine a gamma is some constant that we chose in advance, and this is a bound that holds true for any fixed value of gamma. Later on as we take this bound and then sort of develop this result further, we'll choose specific values of gamma as a [inaudible] of this bound. For now we'll just imagine that when we're proved this holds true for any value of gamma. Any questions? Yeah?

Student:[Inaudible] hypothesis phase is infinite [inaudible]?

Instructor (Andrew Ng):Yes, the labs in the hypothesis phase is infinite, so this simple result won't work in this present form, but we'll generalize this – probably won't get to it today – but we'll generalize this at the beginning of the next lecture to infinite hypothesis classes.

Student: How do we use this theory [inaudible]?

Instructor (Andrew Ng):How do you use theorem factors? So let me – I might get to a little of that later today, we'll talk concretely about algorithms, the consequences of the understanding of these things in the next lecture as well. Yeah, okay? Cool. Can you just raise your hand if the things I've proved so far make sense? Okay. Cool. Great. Thanks.

All right. Let me just take this uniform conversions bound and rewrite it in a couple of other forms. So this is a sort of a bound on probability, this is saying suppose I fix my training set and then fix my training set – fix my threshold, my error threshold gamma, what is the probability that uniform conversions holds, and well, that's my formula that gives the answer. This is the probability of something happening.

So there are actually three parameters of interest. One is, "What is this probability?" The other parameter is, "What's the training set size M?" And the third parameter is, "What is the value of this error threshold gamma?" I'm not gonna vary K for these purposes. So other two other equivalent forms of the bounds, which – so you can ask, "Given gamma – so what we proved was given gamma and given M, what is the probability of uniform conversions?" The other equivalent forms are, so that given gamma and the probability delta of making a large error, how large a training set size do you need in order to give a bound on – how large a training set size do you need to give a uniform conversions bound with parameters gamma and delta?

And so – well, so if you set delta to be two KE so negative two gamma squared M. This is that form that I had on the left. And if you solve for M, what you find is that there's an equivalent form of this result that says that so long as your training set assigns M as greater than this. And this is the formula that I get by solving for M. Okay? So long as M is greater than equal to this, then with probability, which I'm abbreviating to WP again, with probability at least one minus delta, we have for all. Okay? So this says how large a training set size that I need to guarantee that with probability at least one minus delta, we have the training error is within gamma of generalization error for all my hypotheses, and this gives an answer.

And just to give this another name, this is an example of a sample complexity bound. So from undergrad computer science classes you may have heard of computational complexity, which is how much computations you need to achieve something. So sample complexity just means how large a training example – how large a training set – how large a sample of examples do you need in order to achieve a certain bound and error. And it turns out that in many of the theorems we write out you can pose them in sort of a form of probability bound or a sample complexity bound or in some other form. I personally often find the sample complexity bounds the most easy to interpret because it says how large a training set do you need to give a certain bound on the errors.

And in fact – well, we'll see this later, sample complexity bounds often sort of help to give guidance for really if you're trying to achieve something on a machine learning problem, this really is trying to give guidance on how much training data you need to prove something.

The one thing I want to note here is that M grows like the log of K, right, so the log of K grows extremely slowly as a function of K. The log is one of the slowest growing functions, right. It's one of – well, some of you may have heard this, right? That for all values of K, right – I learned this from a colleague, Andrew Moore, at Carnegie Mellon – that in computer science for all practical purposes for all values of K, log K is less [inaudible], this is almost true. So log K is – logs is one of the slowest growing functions, and so the fact that M sample complexity grows like the log of K, means that you can increase this number of hypotheses in your hypothesis class quite a lot and the number of the training examples you need won't grow very much.

[Inaudible]. This property will be important later when we talk about infinite hypothesis classes. The final form is the – I guess is sometimes called the error bound, which is when you hold M and delta fixed and solved for gamma. And so – and what do you do – what you get then is that the probability at least one minus delta, we have that. For all hypotheses in my hypothesis class, the difference in the training generalization error would be less than equal to that. Okay? And that's just solving for gamma and plugging the value I get in there. Okay?

All right. So the second step of the overall proof I want to execute is the following. The result of the training error is essentially that uniform conversions will hold true with high probability. What I want to show now is let's assume that uniform conversions hold. So let's assume that for all hypotheses H, we have that epsilon of H minus epsilon hat of H, is less than of the gamma. Okay? What I want to do now is use this to see what we can prove about the bound of – see what we can prove about the generalization error. So I want to know – suppose this holds true – I want to know can we prove something about the generalization error of H hat, where again, H hat was the hypothesis selected by empirical risk minimization. Okay?

So in order to show this, let me make one more definition, let me define H star, to be the hypothesis in my class script H that has the smallest generalization error. So this is – if I had an infinite amount of training data or if I really I could go in and find the best possible hypothesis – best possible hypothesis in the sense of minimizing generalization error – what's the hypothesis I would get? Okay? So in some sense, it sort of makes sense to compare the performance of our learning algorithm to the performance of H star, because we sort of – we clearly can't hope to do better than H star. Another way of saying that is that if your hypothesis class is a class of all linear decision boundaries, that the data just can't be separated by any linear functions. So if even H star is really bad, then there's sort of – it's unlikely – then there's just not much hope that your learning algorithm could do even better than H star. So I actually prove this result in three steps. So the generalization error of H hat, the hypothesis I chose, this is going to be less than equal to that, actually let me number these equations, right. This is – because of equation one, because I see that epsilon of H hat and epsilon hat of H hat will then gamma of each other. Now because H star, excuse me, now by the definition of empirical risk minimization, H hat was chosen to minimize training error and so there can't be any hypothesis with lower training error than H hat. So the training error of H hat must be less than the equal to the training error of H star. So this is sort of by two, or by the definition of H hat, as the hypothesis that minimizes training error H hat.

And the final step is I'm going to apply this uniform conversions result again. We know that epsilon hat of H star must be moving gamma of epsilon of H star. And so this is at most plus gamma. Then I have my original gamma there. Okay? And so this is by equation one again because – oh, excuse me – because I know the training error of H star must be moving gamma of the generalization error of H star. And so – well, I'll just write this as plus two gamma. Okay? Yeah?

Student:[Inaudible] notation, is epsilon proof of [inaudible] H hat that's not the training error, that's the generalization error with estimate of the hypothesis?

Instructor (Andrew Ng): Oh, okay. Let me just – well, let me write that down on this board. So actually – actually let me think – [inaudible] fit this in here. So epsilon hat of H is the training error of the hypothesis H. In other words, given the hypothesis – a hypothesis is just a function, right mapped from X or Ys – so epsilon hat of H is given the hypothesis H, what's the fraction of training examples it misclassifies? And generalization error of H, is given the hypothesis H if I sample another example from my distribution scripts D, what's the probability that H will misclassify that example? Does that make sense?

Student:[Inaudible]?

Instructor (Andrew Ng): Oh, okay. And H hat is the hypothesis that's chosen by empirical risk minimization. So when I talk about empirical risk minimization, is the algorithm that minimizes training error, and so epsilon hat of H is the training error of H, and so H hat is defined as the hypothesis that out of all hypotheses in my class script H, the one that minimizes training error epsilon hat of H. Okay?

All right. Yeah?

Student:[Inaudible] H is [inaudible] a member of typical H, [inaudible] family right?

Instructor (Andrew Ng): Yes it is.

Student: So what happens with the generalization error [inaudible]?

Instructor (Andrew Ng): I'll talk about that later. So let me tie all these things together into a theorem. Let there be a hypothesis class given with a finite set of K hypotheses, and let any M delta be fixed. Then – so I fixed M and delta, so this will be the error bound form of the theorem, right? Then, with probability at least one minus delta. We have that. The generalization error of H hat is less than or equal to the minimum over all hypotheses in set H epsilon of H, plus two times, plus that. Okay?

So to prove this, well, this term of course is just epsilon of H star. And so to prove this we set gamma to equal to that – this is two times the square root term. To prove this theorem we set gamma to equal to that square root term. Say that again?

Student:[Inaudible].

Instructor (Andrew Ng): Wait. Say that again?

Student:[Inaudible].

Instructor (Andrew Ng): Oh, yes. Thank you. That didn't make sense at all. Thanks. Great. So set gamma to that square root term, and so we know equation one, right, from the previous board holds with probability one minus delta. Right. Equation one was the uniform conversions result right, that – well, IE. This is equation one from the previous board, right, so set gamma equal to this we know that we'll probably use one minus delta this uniform conversions holds, and whenever that holds, that implies – you know, I guess – if we call this equation "star" I guess. And whenever uniform conversions holds, we showed again, on the previous boards that this result holds, that generalization error of H hat is less than two – generalization error of H star plus two times gamma. Okay? And so that proves this theorem.

So this result sort of helps us to quantify a little bit that bias variance tradeoff that I talked about at the beginning of – actually near the very start of this lecture. And in particular let's say I have some hypothesis class script H, that I'm using, maybe as a class of all linear functions and linear regression, and logistic regression with just the linear features. And let's say I'm considering switching to some new class H prime by having more features. So lets say this is linear and this is quadratic, so the class of all linear functions and the subset of the class of all quadratic functions, and so H is the subset of H prime. And let's say I'm considering – instead of using my linear hypothesis class – let's say I'm considering switching to a quadratic hypothesis class, or switching to a larger hypothesis class. Then what are the tradeoffs involved? Well, I proved this only for finite hypothesis classes, but we'll see that something very similar holds for infinite hypothesis classes too. But the tradeoff is what if I switch from H to H prime, or I switch from linear to quadratic functions. Then epsilon of H star will become better because the best hypothesis in my hypothesis class will become better.

The best quadratic function – by best I mean in the sense of generalization error – the hypothesis function – the quadratic function with the lowest generalization error has to have equal or more likely lower generalization error than the best linear function. So by switching to a more complex hypothesis class you can get this first term as you go down. But what I pay for then is that K will increase. By switching to a larger hypothesis class, the first term will go down, but the second term will increase because I now have a larger class of hypotheses and so the second term K will increase.

And so this is sometimes called the bias – this is usually called the bias variance tradeoff. Whereby going to larger hypothesis class maybe I have the hope for finding a better function, that my risk of sort of not fitting my model so accurately also increases, and that's because – illustrated by the second term going up when the size of your hypothesis, when K goes up. And so speaking very loosely, we can think of this first term as corresponding maybe to the bias of the learning algorithm, or the bias of the hypothesis class. And you can – again speaking very loosely, think of the second term as corresponding to the variance in your hypothesis, in other words how well you can actually fit a hypothesis in the – how well you actually fit this hypothesis class to the data. And by switching to a more complex hypothesis class, your variance increases and your bias decreases.

As a note of warning, it turns out that if you take like a statistics class you've seen definitions of bias and variance, which are often defined in terms of squared error or something. It turns out that for classification problems, there actually is no universally accepted formal definition of bias and variance for classification problems. For regression problems, there is this square error definition. For classification problems it turns out there've been several competing proposals for definitions of bias and variance. So when I say bias and variance here, think of these as very loose, informal, intuitive definitions, and not formal definitions. Okay. The cartoon associated with intuition I just said would be as follows: Let's say – and everything about the plot will be for a fixed value of M, for a fixed training set size M. Vertical axis I'll plot ever and on the horizontal axis I'll plot model complexity. And by model complexity I mean sort of degree of polynomial, size of your hypothesis class script H etc. It actually turns out, you remember the bandwidth parameter from locally weighted linear regression, that also has a similar effect in controlling how complex your model is. Model complexity [inaudible] polynomial I guess. So the more complex your model, the better your training error, and so your training error will tend to [inaudible] zero as you increase the complexity of your model because the more complete your model the better you can fit your training set.

But because of this bias variance tradeoff, you find that generalization error will come down for a while and then it will go back up. And this regime on the left is when you're underfitting the data or when you have high bias. And this regime on the right is when you have high variance or you're overfitting the data. Okay? And this is why a model of sort of intermediate complexity, somewhere here if often preferable to if [inaudible] and minimize generalization error. Okay? So that's just a cartoon. In the next lecture we'll actually talk about the number of algorithms for trying to automatically select model complexities, say to get you as close as possible to this minimum – to this area of minimized generalization error. The last thing I want to do is actually going back to the theorem I wrote out, I just want to take that theorem – well, so the theorem I wrote out was an error bound theorem this says for fixed M and delta where probability one minus delta, I get a bound on gamma, which is what this term is. So the very last thing I wanna do today is just come back to this theorem and write out a corollary where I'm gonna fix gamma, I'm gonna fix my error bound, and fix delta and solve for M. And if you do that, you get the following corollary: Let H be fixed with K hypotheses and let any delta and gamma be fixed.

Then in order to guarantee that, let's say I want a guarantee that the generalization error of the hypothesis I choose with empirical risk minimization, that this is at most two times gamma worse than the best possible error I could obtain with this hypothesis class. Lets say I want this to hold true with probability at least one minus delta, then it suffices that M is [inaudible] to that. Okay? And this is sort of solving for the error bound for M. One thing we're going to convince yourselves of the easy part of this is if you set that term [inaudible] gamma and solve for M you will get this. One thing I want you to go home and sort of convince yourselves of is that this result really holds true. That this really logically follows from the theorem we've proved. In other words, you can take that formula we wrote and solve for M and – because this is the formula you get for M, that's just – that's the easy part. That once you go back and convince yourselves that this theorem is a true fact and that it does indeed logically follow from the other one. In particular, make sure that if you solve for that you really get M grading equals this, and why is this M grading that and not M less equal two, and just make sure – I can write this down and it sounds plausible why don't you just go back and convince yourself this is really true. Okay?

And it turns out that when we prove these bounds in learning theory it turns out that very often the constants are sort of loose. So it turns out that when we prove these bounds usually we're interested – usually we're not very interested in the constants, and so I write this as big O of one over gamma squared, log K over delta, and again, the key step in this is that the dependence on M with the size of the hypothesis class is logarithmic. And this will be very important later when we talk about infinite hypothesis classes. Okay? Any questions about this? No? Okay, cool. So next lecture we'll come back, we'll actually start from this result again. Remember this. I'll write this down as the first thing I do in the next lecture and we'll generalize these to infinite hypothesis classes and then talk about practical algorithms for model spectrum. So I'll see you guys in a couple days.

[End of Audio]

Duration: 75 minutes