in this lecture i'm going to talk about how we achieve our actual goal in machine learning which believe it or not is not to minimize the training loss so we want to build a predictor that looks at the training data but we wanted to do well on future data of a similar form if we're classifying images we want to be able to classify images other than just the images in our training set so we're looking for something that will perform well on new data previously unseen as well as if we see the training data again right so that's our real goal we call this generalization performance on new data now a key question is does minimizing the training loss the error on the training data achieve this so i ponder that for a minute stare at the training data do as well as possible does that cause us to do extremely well on new data or as well as we can do um i learned uh you probably have your own real life lessons about this issue but i learned a version of this one and semesters when i collected pictures of all my students with their name written on the back to try to learn their names or you could do it digitally you know with their anyways if you have a collection of images of your undergrads and the purpose is to learn their names you can study those pictures and it will help you learn their names for sure but i discovered that if i there's only so much accuracy i could get that way and if i kept studying i would learn what shirt they were wearing and what background they took the picture against what hat they were wearing that day and of course i would be very good if the student would present themselves in the same shirt and hat against the same background but it was unclear that that made me better if some other student showed up in that hat or shirt or background i might mislabel so minimizing training loss was not serving me very well at that point then you may have your own lessons at this point and then there's also sort of the thought experiment you can do of what if i build this predictor consider a predictor that it takes a training set there's a training set here x i y i i equals one to n that's and training points feed it in to do the learning and all it does is remember it and when it needs to classify some new item x it looks to see if it's in the training set and returns that value so if it's the if x that comes in is x i in the training set it returns y i and if it's not in the training set it just generates an error now this is a terrible predictor for generalization it does no generalization at all but it gets zero training loss so that's the point of this example is to make it very clear that training loss is not our goal to shed more light on this we want to think about something called bias and variance in our learner so we have something called the bias variance trade-off to understand what these are realize that you could imagine that we have a data source that's generating our data if we don't then i don't know why we need to generalize to more data there won't be any more right we have some source of data it didn't just generate the training data and die it's going to generate more data maybe it already has and we want to be able to classify that more data so if we go back to that source more than once and draw different training sets we could run the learner twice or many times right and so we imagine our data source is a probability distribution over these x y pairs and we're sampling from it and we can sample different training sets and so we want to ask how our learner behaves over this random process and that allows us to define the concepts of bias and variance so bias is going to result from building into our learner some idea about what it should and shouldn't do other than minimize training error so a typical thing would be you can't represent arbitrary functions maybe you can only represent quadratics or linear functions and also functions of what functions of certain features you can build in certain features when you're designing the learner as you do select features and once you've selected features you've sort of forced the learner to learn some function of those features and not some other function and as a result of this the learner is in in many cases of actually completely capturing the underlying data source maybe the underlying data source is generating patterns that are not linear combinations of those 50 features you picked and so it cannot capture them and that means that we expect to get error at the end so the test error here is we run this experiment sample data run our biased learner get a learned function and then the same data source generates test data and we measure the the error or the loss or some something like that of a predictor how good a predictor did we get we use these same test data to evaluate the results of the second experiment but it has different training data sampled from the same source that different training data may learn a different function but over this whole process we expect to see test error on that you know different sampling issues here may average out over this process but even over that averaging we expect to see test error because we can't necessarily even represent the underlying data source because we were biased we did not build our learner so that it could learn any function at all now a key thing that and that sounds bad we don't want bias and bias in an intuitive context is in fact thought of as a negative thing you shouldn't be biased right but but and i should note this is not the same notion of bias as but it's a related notion as we use when we're talking about human affairs even though there is an ai notion of that say being biased against um against certain minorities in doing image classification that's not what this bias is i mean certainly if you built in a a bias in in the learner against minorities that would be this kind of bias but what's more typically the problem in ai or also the brahman ai is that the data source itself is generating bias training data that's not what this bias is about this bias is about the learning systems designer limiting what the learner can do or or distorting what the learner can do to try to make sure that the learner learns a certain kind of function and that function may not fit the data source and that gives you expected test error now you can also get expected test error from another source and that is the data source itself may not consistently label right it may give you the same x 90 of the time as a cat but 10 percent of the time it says it's a dog and if the data source is doing that that's irreducible error nothing you do when the learner is going to get that test error out so your total expected test error is not just the bias it's the bias and the irreducible error but the bias is what's of interest to us today bias sounds bad but bias is going to be required to get generalization so i want you to think about when um when when this learner when this learner is learning from this data set why should it care what happens with this data set it's a different data set it's focused on this data set learner up here is looking at d1 but if we want a consistent answer between f1 and f2 then this learner up here needs to have some kind of care about what its fictional twin is going to do if it has no care at all then it may as well generate an error whenever it sees data that might have been in d2 it has no information at all about what should happen for data outside of d1 unless we build in a bias that somehow connects the two just the idea of saying similar data should get similar answers is a bias it's not necessarily true of all data sources so this learner if it's saying i want to learn a function that gives similar data similar answers it's going to choose a function that may actually do something that has higher training loss on d1 in order to be smoother all right so that's the nature of bias is that this learner is concerned with more than just training loss on d1 and this one's concerned with more than just training loss on d2 and as a result they both learn similar functions because they have an eye to say the smoothness of the answer so when my chemistry teacher is telling me to draw a straight line through the data my chemistry teacher is biased towards a linear answer or a smooth answer for the experiment and it's saying give an eye to the fact that you haven't seen all the data and other data may fall in between and we want to give similar answers so draw a smooth line that's bias in particular okay so we need bias for generalization it absolutely required we will not get this generalization if we don't build some kind of bias into our learner otherwise it may as well just memorize the training set and the problem with memorizing the training set is we get high variance if we if we don't build in bias we're going to get variance and we we will inevitably get some variance what is variance variance is the variation of these different predictors f1 f2 the predictions they're making how much does their ver their prediction vary around the expected prediction bias is the expected prediction is going to be wrong but even around the expected prediction different runs of this may give very different predictions for the test data and if they do they have high variance that's clearly an undesirable thing that that we run our learner on the data set we get a predictor somebody else runs the same learner on a different trading set but from the same data source they get a predictor and the two predictors if they're wildly different that's high variance they can't both be right on any given data point so one of them is badly wrong if they have high variance and so basically this constant of variance corresponds to high sensitivity to the choice of the data set so t2 and d1 give very different learners these learners are doing things like memorizing all the data and that's the corresponding thing in my chemistry teacher's classroom of connect the dots when i plot the curve from my experiment instead of a smooth line i connect the dots memorize the data hyper focus on the data set and that's high variance the way we reduce that variance is to build bias into our learner the bias says don't necessarily minimize the training loss don't memorize the training data instead smooth it out or whatever i want to be biased towards do something that i'm biased towards i give a high prior weight or a probability of a smooth function or a low dimensional learned function or or perhaps even linear or one that can be described with few features i need to build some kind of bias into the learner to give it any hope of generalizing and not just memorizing the training data that's the topic of this lecture so the basic problem that results if you do not bias your learner is called overfitting probably many of you have heard the term it refers to over focusing on the training data that is over overly tightly fitting the training data and it occurs when insufficient bias when we have not enough bias leads us to hyper focus on the training data and get higher variance and consequently we've overfit we've gotten poor test performance because of it so that's a motivation to have bias does that mean bias in a machine learning context is a good thing well the fundamental problem is that there's a trade-off and too much bias can in fact lead to poor test performance this means we've forced our learner so far in the direction of what we might think is a smooth or simple hypothesis that it can't represent the patterns at all not only can't represent the noise in the training data or the sampling error it can't even represent the fundamental patterns so yes too much bias will lead to failure to focus enough on the training data we won't fit the trading data patterns we can't represent them if we buy us a learner too much so in this example it should be clear that we maybe could learn something like this some kind of nice division like this and if we're the fundamental question is is what if there was say more going on there was maybe a dot here what do we want to have the learner do is the learner if the learner is hyper focused on the training data it might learn some squiggly boundary and maybe that's what we want and maybe that's not what we want it's possible that this is more of a a second degree you know quadratic and that's a good concept here it depends on how much bias we put in whether it will be able to capture that point or not there's not an obvious and immediate right answer to whether this is a quadratic function or a linear boundary that simply gets this point wrong if you minimize training loss you're probably going to get something that's very jagged so there's just a trade-off between bias and variance it's not that more bias is always good or less bias is always good so that's going to lead us to have to find a system for choosing how much bias to build in but before we can talk about that we need to talk about how we build bias into our typical machine learning systems in practice so to build bias in we want to understand bias is a technique we used to sort of force simpler solutions simpler can mean many things though what kind of a notion we consider simpler depends a lot on the context so this is a design choice that can be a bit of an art in most cases it's being bias is being used to do something that we expect in science where we prefer the simplest solution this is again with my chemistry teacher the line is much simpler than connect the dots and so we view it as a more likely um true result or more right likely accurate representation of the underlying chemistry phenomenon that i was experimenting with and that the variations from the smooth plot were probably noise that is something called occam's razor where we have a preference for a simpler explanation explanation probably a more perfect word than solution here a machine learned predictor is an explanation of sorts of the patterns in the data and we would like to prefer the simpler ones so there's three general classes of very practical technique that are used the first is simply to to choose fewer features so limit the hypothesis space we might say it's going to be a linear combination of our features and we're not necessarily going to pick as many features as we could now that can be a very powerful way to limit the complexity of the hypothesis and force generalization but you've got to realize that if you limit it too much and you don't include features that let you express the key patterns you want you will you will have a terrible problem fitting the data you'll have a very high error due to the bias that's where we'll get this too much bias leads to poor test performance so and in many of our especially deep learning situations we really don't know what we want the features to be and we certainly can't say we're going to limit it to this or that so um this can be a challenging thing to do to choose to limit the features sometimes we can do it in in a deep neural net setting by the way this limiting the hypothesis space could be limiting the structure of the network designing the structure of the network a certain way so that it won't it can only learn things that fit that structure that's a big deal in neural deep neural net design is designing the structure so limiting hypothesis space is still a big deal even if you can't pick features ahead of time you could pick your feature learners which is essentially what's happening in deep neural nets by setting up the structure of the deep neural net so that's limiting the hypothesis space very important way of biasing your learner a second very important way very standardly used is to penalize complicated weight vectors so even if you have a giant number of features you could essentially require the weight factor to be mostly zeros don't use most of the features most of the time require a simple explanation that taps only a subset of the features where a slight generalization of that is we'll let you use values on the features but just don't let the weight values be very much larger than zero most of the time so we're going to take the norm of the weight vector and penalize it basically when we measure the loss in gradient descent we're not just minimizing the training loss we're also required to minimize this penalty by choosing a weight vector that's sparse or mostly sparse this is a soft version of sparse all right so we just do gradient descent but with a more complicated loss function this is called the regularization term it's penalizing complexity so that's also widely used again this uh lambda w squared the square you know it turns into lambda w when we take the gradient okay and then the simplest thing and also widely used way to avoid a complex overfitting of the training data is to simply stop training don't don't train all the way until it is converged just head down the gradient for a certain length of time number of iterations and then stop early stopping of training it prevents overfitting in a number of ways among other things it tends to prevent the norm of the weight vector from growing too large if you start your weights out at zero they can only grow so much in so many iterations and this when to do the early stopping is a hyper parameter of the technique it's a parameter that we're not optimizing in the learning but we need to choose it before we do the learning like the gradient descent is optimizing just the weights the other parameters are called hyperparameters which features we use as a hyperparameter what value of lambda we use as a hyperparameter and how many iterations to do for early stopping is a hyperparameter and so one of the things we'll see next is how we would set these hyper parameters which basically has to do with the broader question how do we know how much bias to use how do we know how to set this up so we have the right amount of bias to do that we have to get some kind of estimate of how our technique is working to get such an estimate we need to be able to detect that overfitting is happening okay the first step is to be able to detect it in our result well this is just being able to measure the quality of our result we don't train on the training set and then look at there we got on the training set and then say that's how much success we had because as we've seen that's not our goal to do well on the training set so we need a separate set of data a different set of data than the one we trained on that we don't look at and don't touch and that's called the test set and after we're done with our learning we can run the final predictor that we got on the test set and see how we did and that will give potentially much worse answer than the training loss that we achieved in gradient descent if there's a big gap there then we suffered from uh overfitting we're always going to suffer from some forms of error in the direction of overfitting the test set is always going to give worse performance than our training but we want to reduce that and we can change our choices of all these hyperparameters the features the regularization the early stopping we can change all of that to see if we can improve this performance but if we do that we'll be basically milking the test set to try to set those parameters and it will no longer be a good test set to see how we did in the end so we don't do that we don't use the test set for that purpose instead we introduce one more thing which is called the validation set validation set is a another bunch of data that we don't train on so we hold back typically 10 of the data something like that don't get too wetted to that number hold back some of the data what you can afford to to hold back and don't use it in your gradient descent okay so this is for detecting during training detecting overfitting to guide key choices so we're guiding the hyper parameter choices using the validation set so we reserve this away and we use it only as an estimated test error so that's weird we have the test set we could estimate test error there but we're holding that back to not touch it at all during our hyper parameter training the hyperparameter draining we're using the validation set and that taints the validation set now we know we've set the hyper parameters to do well on the validation set but we don't know yet uh if it's really if we really over fit now the validation set as well or if we are truly generalizing to the test set so i want to hold the test set back okay let me try to get the the the process a little clearer for you we have three sets training set validation set and test set validation set and test set we're not going to touch at the beginning we just train on the training set we'll use some settings for our hyper parameters we're going to have to choose some features choose some some lambda if we're using regularization choose early stopping if we want make those choices and do some training see what training area we get play around with those choices get the training error low not touching the validation and test set just get the training arrow okay now we've probably overfit so now check our performance on the validation set and see that we've already fit now back away some of these choices and adjust them to try to get the the training on the training set to generate good performance on the validation set so you're not doing gradient descent on the validation set all you're doing is peeking at it after you do gradient descent to see if you did well on it and if you didn't do well on it you need more bias so add more bias in here train on the training set again and then take the resulting learner and test it on the validation set repeat that cycle train on the training set test the resulting learner on the validation set and what are you changing as you go around the cycle you're changing the amount of bias increased bias until the validation set performance looks good it's always going to be worse than the training set but it can look decent you have to use some judgment whether you've um whether you've achieved a good enough performance on the validation set without totally cheating on the validations that you don't want to you know peek into the validation set so intensely that you really designed these features especially for it it's just a measure you really shouldn't be looking at the data in the validation set at all just the performance measure and try to drive it down by making hyper parameter choices in your meanwhile you are looking at the data in the training set hand examining it to see what features you might want to use or not use and so forth okay so you're going to have a design process that melts the training data as much as it can without damaging the validation error and that that's going to be development loop we'll look at a diagram of in a minute and after you finish you decide you can't get the validation error down anymore without cheating and peeking into that data set and training gradient descent on it then you're done you've got your final predictor now you use the test set to see how you did that's all you use it for see how you did and if you did badly that kind of is unfortunate but that doesn't mean you can now go iterate this again you can but the more you do that you run into this problem that you if you use the test set in this development loop it no longer provides a good estimate of your quality of your result because now you're have the danger of overfitting the test set that's why we have the three test set is just not touched during during this development during training or development okay so the final summary overview picture of all of this this is um as nice as someone's summarized on this slide from the stanford course that has been the inspiration for some of our work um and that is this overall algorithm that we start by splitting our data into these three test sets this valve is for validation and then you're going to need to examine the data and try to get an understanding of what might work so you can choose features and how many features and tune the hyper parameters then you're going to run whatever learning algorithm gradient descent loss that you've chosen and take a look at the results what are the training and validation what is so notice the learning algorithm is running on the training set it's ignoring the validation set this does not use the validation set but then you'll take a look at what resulted the error rates on the training set hopefully are getting quite low if they aren't you just don't have enough features you have too much bias if you can get the training error low then your question is what happened to the validation set that you didn't look at in designing this this algorithm and you didn't look at it in the gradient descent and setting the weights and so you're going to see what if that error rate it resembles the training set weight then you're doing pretty well but if the validation rate is much worse than the training weight you might be inspired to tune hyper parameters add or subtract features and and so forth increase the bias so that it's not able to fit the training data necessarily quite as well but we'll get better validation there so there's a brainstorming black art element on this loop when you're done you can see how you did it's like getting your quiz graded you don't get to take it again you take it once you look at the test at once at the end okay so that's uh that's our coverage of overfitting the bias variance trade-off you increase bias to lower variance if you have too much bias it will lower your performance in the end