in this lecture we're going to discuss a more advanced form of classification learning uh more a bit more principled in terms of statistical models called logistic regression logistic regression addresses a problem such as you might call it a problem in our basic classification scheme which i have diagrammed here in our basic classification scheme we're given something to classify and a supposedly correct label for it from our data source so this is a pair from the training data and we process it and we get this margin the margin as we said last time is some measure of how correct we are how correct we are on this data point so it's a real number if it's negative we're we're the opposite of correct how negative it is is how wrong we are and how positive is how correct we are and this uh in this box is just multiplication the score is multiplied by the the right answer notice that this is plus or minus one this was just this was the topic of last lecture so what is the problem the problem is this fuzzy notion of how correct what does it mean a real number is how correct all we have really is that if it's more positive we like it better but we don't have a meaning for that and logistic regression attempts to remedy that by saying well what we really want to know is the probability that this is the right answer so we want to we want to view our our prediction as a probability and so when the margin is positive we want it to be a high probability that y i is the label and when the margin is negative we want it to be a low probability so it leads us to this idea a predictor should give us the probability of a given label given the training input so given that we've seen the input x i the predictor should tell us what is the chance that the label should be y i and it should tell us that in this for plus one and then one minus that will be the the probability that the label is minus one of course we have the correct answer on the training data but but we want to get an answer i want the predictor to give us an answer for for that without seeing why i so you will notice then that the answer it's giving us is no longer a plus one or minus one it's giving us a probability that it's plus one yeah so maybe it would be better to write this as plus one so the probability that y equals plus one given x i that's what we want to have a prediction of now that's a nice idea sort of conceptually that we would like to get a probability but notice that everything throughout here the score the margin the margin we can't even that's not our prediction that's got to do with evaluating how well we did because we need the answer to compute the margin but the score as well is a real number an arbitrary real number it's not the probability probability is something between zero and one right so a fundamental issue is that the the range of our dot product isn't 0 1. so that's a fundamental problem we need to address score ranges over the real numbers not 0 1. moreover i use square brackets as i write this range here i want to point out that throughout ai and handling uncertainty and particularly machine learning we're really not going to be very attracted to using the end points we never want to say there's absolutely no chance of the other answer we're rarely going to want to say that now that literally leads into the brittleness that deterministic ai faced over decades always allowing some however miniscule probability of anything okay so but here we have this problem that our score that is fundamental to our prediction like the only thing we've learned how to do is adjust weights in a linear combination and that gives us a score that ranges over the real numbers so how do we get it into 0 1 well of course naturally we are going to be thinking about a function like the logistic function the sigmoid function that you have already met and we'll introduce it here as well because that's going to map the real numbers that's going to give us a bijection from the real numbers into 0 1. so not only does it map the real numbers into zero one but each real number goes to a different place so that mapping has an inverse so it's basically a one-to-one connection between the real numbers and zero one so it's a nice function in that regard but it's also naturally motivated and i want to addre i want to derive that motivation here rather than just jump to the sigmoid function so what we've arrived at though is we can't use the score as the probability we can't imagine that the score is the probability it's not even the right range so what could the score be probabilistically what probabilistic quantity or idea to answer that i want you to think about things that come up in probability that are not between zero and one and in particular a very natural thing that comes up when we're saying that say a weighted coin is three to one heads there's a three in there that's not between zero and one three to one heads represents a point seven five probability of being heads or it could be one to three and then it's a 0.25 probability of being heads and what these are these ratios they're odds the ratio of the probability of one value over the probability of the other value so i'll just put that example here odds of heads three to one is the probability of heads equals 0.75 so there's two different ways of saying the same thing but

the first way the odds can take on arbitrary high ratio or low ratio it's always positive but it can take on the odds would be three to one or one to three we can have anything all the way down to but not reaching zero or arbitrarily large as a ratio the way we get it to reach the whole real numbers is to take the log of the odds and now that log can take on any real value but there's another reason to want to take to to consider the real numbers and that is when you think about odds you're typically doing i mean there's another reason to think about taking the log and that is when we we think about odds we're usually doing something like this 1 to 1 10 to 1 hundred to one a thousand to one ten to the fourth to one ten to the fifth to one one in a million no one ever says 1 in 999 984 and if somebody said that you'd be like well yeah one in a million right um we're interested typically in the magnitude of that exponent um one in a million one hundred 000 natural things to say one in 225 000 we'd only said if our data happened to give that number and then we'd be thinking oh it's between one and a hundred thousand and one in a million um we naturally think about probabilities on a log scale so that's a second reason to consider taking the log so that leads to the idea that the score represent the log odds of plus one given the input x so given the input there's some probability of plus one being labeled we're not going to say it's ever certainly one or certainly zero with some probability and we're going to let the score represent the log odds now i don't really get to say what the score represents i can wish for it to represent something but how do i get it to represent that like how in designing the system can i make the score represent something the only thing that if i have any influence on here is what i'm training the weights towards weights are going to get drawn to some convergence whatever they converge to that that's going to be what the score represents and so i need to influence that what influence influences that thinking through the process what influences where the weights go is what loss function i use i'm going to minimize the loss function so to make this work the way i'm saying i don't just get to say the score represents the log odds of y equal to plus 1. i have to choose a loss function that makes it represent that and we're not anywhere close to doing that yet in this lecture all i've done like picard to say make it so that doesn't work here i have to choose the loss function to make it so so i want to choose a loss function such that if that loss function becomes zero if i have no loss my score will actually be the log outs and we're not we haven't developed an understanding here of how we would do that so that should seem very mysterious at this point the way we're going to get at that is we're going to make sure that the prediction the prediction is the probability and we're going to divine loss in terms of that prediction so we we we already said we want the prediction to be this the probability of y and i haven't got that yet all i've got is suppose my score was the log odds well let's make this key claim at this point claim if score is log odds then sigma of score is the probability y equals plus 1 given x as desired where sigma is the logistic function we're going to see in just a moment exactly what that is let's prove this claim and improving it we will derive what the sigma function needs to be so let's use s for score if square is the log odds that means the natural log of p over 1 minus p here i'm going to call this p and this s okay so the odds is the probability of heads over the probability of tails in this case plus 1 and minus 1. so p and 1 minus p and the log odds is the log of that so so this is our statement the score is the log odds from this i want to derive my claim it's pretty straightforward to do so i just need to solve for p which you should have the tools to do i'm going to take the natural exponentiation on both sides and get oops i want to do that in black get e to the s e to the score equals p over one minus p plus one minus p e to the s equals p and then i can distribute e to the s minus p e to the s equals p e to the s equals p times 1 plus e to the s p equals e to the s over 1 plus e to the s equals 1 over 1 plus e to the minus s equals the sigmoid function of s that is the sigmoid function right there so doing some very simple algebra i can solve this claim that the score is the log odds for the probability of plus one if the score is the log odds that is the probability of plus one over the probability of minus one take the natural log if that's what the score is which it isn't yet but if it is then the probability itself that i'm looking for is sigma of the score so this is what i'm going to take to be the prediction in this classifier this classifier doesn't predict plus one or minus one it predicts a probability of plus one which in a way is more useful you can always say if it's point five or larger i'll say the prediction is plus one you can always threshold it but it's going to give you a natural probabilistic notion of confidence and notice that if i say i'm three quarters confident that it's plus one we can check that by testing me over and over and i should be getting three quarters of them right when i'm three quarters confident so we can measure whether i'm empirically getting what i'm claiming all right so let's

see so this gets us to the last piece of this idea which is still a pretty significant piece to develop here we're not all that close to done but the last key piece is design the loss to be minimized that is to be zero when sigma of the score is the empirical or is the data sources probability of y equals plus one given x so if we were to draw from the data over and over we would get this probability that this the the sigma of the score says we're going to get we'll get some plus ones and some minus ones will get the right fraction and when those match the loss should be zero now i say the data source but in practice it's going to be the training data not the actual data source we'd like those to be the same thing right we like the training data accurately capture the probabilities in what the data source is actually generating asymptotically but in practice of course there'll be noise and it won't necessarily match but but the loss is going to be something that's measuring how close we are to capturing the probabilities in the training data so if the training data only show heads then this probability is one in the training data even if it's possible to get a tails or minus one never came up in the training data then empirically we are seeing it as plus one and the loss will not be zero unless the score says it's plus one okay so this gives us a remaining task which is to notice that well okay to formulate this task notice that if we look over all the different x's and y's this is representing a probability just the sigma of the score is representing a probability distribution um for each x that's representing a binary probability distribution probability of heads probability of tails this is and so we need and in more general more generally we can have we can use the same kind of technology to generate a probability distribution over more than just heads and tails over cat dog bird whatever so we can have a multi-class classification where we're using multiple scores to generate a probability distribution so the same ideas will come up there in this lecture we just got heads and tails and so we just have one score and sigma of that score is the probability of plus one which i'm loosely calling heads in order to design such a loss function we need to think about distance between probability distributions how do we measure distance between probability distributions so for this we have two distributions the p and the q distribution and we want to know what's the distance between them here they're conditional distributions because we've got a training data point we're labeling so we can sort of ignore that and imagine it's fixed we have a particular training data point we're labeling and the training data or our source gives a certain chance of plus one whereas our prediction gives a possibly different chance of plus one and how do we wanna measure that distance what do we wanna say the distance between those two is so that the distance is zero when the probabilities are the same and to to quantify that we're going to reach for a little information theory which is also coding theory this we're going to define h of p equals the entropy of p which is the expectation when we draw an x from p of negative natural log the probability of x now that's if you haven't seen this kind of study of probability distributions we're not going to spend a lot of time on it here and this is going to remain a little mysterious to you it is very interesting um and you can dive into it if you want and learn more about it we don't need to do all that much with it in this class but there's plenty of relevance if you were to take more time we have claude shannon a brilliant mathematician in the um early mid 20th century developing coding theory and information theory and the shannon coding theorem this will tell you that this value h of p gives the expected number of gnats and that's a term you may not know think of it as bits except that base e something that will seem a bit mysterious to you because we took the natural log if we used a log base 2 there we would get bits expected number of gnats needed to communicate a draw from p so the distribution p we're we're drawing heads and tails from it and i draw a head or a tail and i want to communicate to you what i drew and i'm going to need on average h of p nats if we use natural if we use blog base two we'll say bits i'll need h of p bits to communicate to you what i drew under the best code now this may seem mysterious uh i'm sure it seems mysterious let me just shut a little bit of we're not going deeply into this but i want to shout a little bit of light on on it why would it take any less than a bit to tell you whether i drew a heads or tails that seems a little contradictory let's just switch to log base 2 and bits because talking about gnats is an extra layer of confusion imagine this was log base 2 and we had bits here obviously if i have if i can send you a bit i can tell you whether i got heads or tails so i definitely don't need more than a bit to tell you that and what this will tell you this quantity will tell you is that if i use log base 2 here and i have a fair just coin distribution probability of heads is one half this is going to evaluate to 1 and i will need one bit to tell you i'm flipping in my fair coin over and over and i'm sending you a bit every time and you're finding out but now the situation you'll be forgetting

here if if that's all you think that's all that's happening is suppose my coin was heavily weighted and it was going to come up heads nearly all the time well now i can package together my my flips and say well i'm going to flip it 10 times and i'm going to send you a one if they're all heads if they're not all heads i'll send you with zero followed by a 10 bit code to send what they actually were so i'm going to send you 11 bits if they're not all heads and only one bit if they're all heads now you should see if the probability of heads is high enough i'm going to save on this because i'm going to usually just send you one bit yeah it was all heads again yeah it was all heads again so every 10 flips i send you one bit i'm saving enormously if you know if only once in a millennium we get the opposite so it's it's um it's like this at the end of the year i will send you a bit that tells you that 365 times the sun came up or whether i could see it or not it came up in the morning it's behind the clouds still came up i'll send you a a one at the end of the year if the sun came up 365 times if it didn't come up 365 times i'll send you a bit code of 365 bits tells you which days it came up on and which days it didn't so i'm going to pay a pretty big price on those years when there was a day when the sun didn't come up however that never happened so i'm just going to send you one bit every year it's a lot less than one bit per day and so what this being pointed out here is that a low entropy distribution where where it's almost all on heads and and almost certainly heads doesn't need as many bits to communicate draws from the distribution as a high attribute distribution the highest entropy distribution is totally random totally uniform so this is a measure of how much we can save by exploiting the weightedness of the distribution and this is a measure we can use to start talking about how far away another distribution q is from b and that's why i'm talking about it so now i'm going to use this definition this kind of definition to talk about how how two codes relate to each other so now i have two co two probability distributions not two codes two probability distributions p and q i'm going to talk about the cross entropy of p using a code optimized for q or just the cross entropy of p and q so this is the concept here is i'm sending a message about what happened when i drew from the distribution p but i'm using the code that was optimized for a different distribution so if if these distributions are far apart i'm going to get a bad result here let's say i'm trying to send you a message about whether the sun came up or not i'm using that code that i weighed 365 mess flips and then i send you a one if if the entire 365 were all ones but otherwise i have to send you a 366 bit code and i'm using that but what i'm actually getting is coin flips i'm not getting a reliable heads heads headsets i'm getting 50 50. half the days the sun comes up half the days it doesn't and i'm trying to code that using the code for the sun that always comes up and that's going to result in a really bad mismatch i'm going to send you 366 bit codes every year instead of just the one bit that i was aiming at and so the cross entropy will be high because i'll be drawing from p but i'll be using a code optimized for q and so what is this mathematically it's the expectation when we're drawing from p of negative log q the probability under q of x so basically my message cost is being paid in the q distribution even though i'm drawing from the p distribution and if you believe the shannon coding theorem that the best possible code for for drawing from p is measured by negative ln of p of x you'll realize that this is minimized when p and q are the same distribution otherwise the coding theorem wouldn't be true okay so this is going to be our motivation for using negative ln of our predicted probability as our loss it's called cross entropy loss i want to note that this this is minimizing p equals q but it isn't zero when p equals q so i didn't actually achieve this at the top i'm going to achieve minimized and i'll do a little bit of an aside on that because it's something you may encounter or may have encountered and that is what's called the kl distance between distributions k and l were people kobach and liebler but everybody calls it kl distance that's going to be h p of q minus h of p what we're saying is there's some sort of basic cost to send a message about a draw from p and there's an additional cost if it's miscoded and that additional cost is the distance but this this cost is the additional cost plus the basic cost so i have to subtract off the what would have been the minimum and figure out how much of a penalty i paid for coding at sq that penalty is the kl distance this is what i really want to minimize because this is zero when they're the same however when i'm changing the weights i'm only changing q so this is just a constant that i don't need to involve in the calculation if i want to find the minimum value i don't care if the minimum is at zero or some higher value i want to change the weights to minimize this then i'm going to be minimizing hp of q by changing the weights and i don't need to have a loss function that reaches zero i just need one that reaches a minimum when this would be zero so i don't actually have to compute this underlying

entropy of the data source i just need to know what the difference is between my training data and my predictions and that's this term that's this right here so it's going to motivate this definition cross entropy well i need to put it in black let's see this is a very famous loss function ubiquitous in machine learning you'll see it if you stay in machine learning you will see this loss function over and over and over and it's motivated by these definitions it's minus ln sigma of the score so this was our prediction our predicted probability and the natural the negative natural log of it is the penalty that we're paying for if we were communicating our message in this using this probability instead of the actual probability empirically in the training data and that's right here that's the negative ln of the predicted probability and the way we get this expectation is this is being computed over the training data this is the training data draw so we're drawing from the training data and averaging this over all the training data so that's this expectation and what are we averaging over the training data the loss function so we then have that this loss the cross-entropy loss is minimized when our predicted probability matches the training data's probability of getting a plus one given x that's when we'll reach the minimum and when we achieve that minimum these match then we will have if we if we if we actually get a perfect match there the score will be the log odds now we're not going to get the score to be the log odds because we're not going to be able to perfectly match the training data there'll be trade-offs between matching one training data point matching another given the limitations of our hypothesis class and our feature extraction and so forth so we're not actually going to get the score to be the logouts but we're trying to make it the log odds that's what our training is doing if sigma's of score could give the exact empirical probability of y given x at every at every x and the training data actually had had enough data to define that well then we would actually have the score being the log odds okay and then it won't surprise you that we are in fact going to use stochastic gradient descent to identify the local minimum of this loss function however you identify the local minimum we will call it logistic regression but in this class we will imagine we're using sgd or gradient descent and that is what we will call logistic regression a classification technique that gives you a probability of being in your class