MachineLearning-Lecture05

Instructor (Andrew Ng):Okay, good morning. Just one quick announcement and reminder, the project guidelines handout was posted on the course website last week. So if you haven't yet downloaded it and looked at it, please do so. It just contains the guidelines for the project proposal and the project milestone, and the final project presentation.

So what I want to do today is talk about a different type of learning algorithm, and, in particular, start to talk about generative learning algorithms and the specific algorithm called Gaussian Discriminant Analysis. Take a slight digression, talk about Gaussians, and I'll briefly discuss generative versus discriminative learning algorithms, and then hopefully wrap up today's lecture with a discussion of Naive Bayes and the Laplace Smoothing.

So just to motivate our discussion on generative learning algorithms, right, so by way of contrast, the source of classification algorithms we've been talking about I think of algorithms that do this. So you're given a training set, and if you run an algorithm right, we just see progression on those training sets.

The way I think of logistic regression is that it's trying to find – look at the date and is trying to find a straight line to divide the crosses and O's, right? So it's, sort of, trying to find a straight line. Let me – just make the days a bit noisier. Trying to find a straight line that separates out the positive and the negative classes as well as pass the law, right?

And, in fact, it shows it on the laptop. Maybe just use the screens or the small monitors for this. In fact, you can see there's the data set with logistic regression, and so I've initialized the parameters randomly, and so logistic regression is, kind of, the outputting – it's the, kind of, hypothesis that iteration zero is that straight line shown in the bottom right.

And so after one iteration and creating descent, the straight line changes a bit. After two iterations, three, four, until logistic regression converges and has found the straight line that, more or less, separates the positive and negative class, okay? So you can think of this as logistic regression, sort of, searching for a line that separates the positive and the negative classes.

What I want to do today is talk about an algorithm that does something slightly different, and to motivate us, let's use our old example of trying to classify the team malignant cancer and benign cancer, right? So a patient comes in and they have a cancer, you want to know if it's a malignant or a harmful cancer, or if it's a benign, meaning a harmless cancer.

So rather than trying to find the straight line to separate the two classes, here's something else we could do. We can go from our training set and look at all the cases of malignant cancers, go through, you know, look for our training set for all the positive examples of

malignant cancers, and we can then build a model for what malignant cancer looks like. Then we'll go for our training set again and take out all of the examples of benign cancers, and then we'll build a model for what benign cancers look like, okay?

And then when you need to classify a new example, when you have a new patient, and you want to decide is this cancer malignant or benign, you then take your new cancer, and you match it to your model of malignant cancers, and you match it to your model of benign cancers, and you see which model it matches better, and depending on which model it matches better to, you then predict whether the new cancer is malignant or benign, okay?

So what I just described, just this cross of methods where you build a second model for malignant cancers and a separate model for benign cancers is called a generative learning algorithm, and let me just, kind of, formalize this. So in the models that we've been talking about previously, those were actually all discriminative learning algorithms, and studied more formally, a discriminative learning algorithm is one that either learns PFY given X directly, or even learns a hypothesis that outputs value 0, 1 directly, okay? So logistic regression is an example of a discriminative learning algorithm.

In contrast, a generative learning algorithm of models PFX given Y. The probability of the features given the class label, and as a technical detail, it also models PFY, but that's a less important thing, and the interpretation of this is that a generative model builds a probabilistic model for what the features looks like, conditioned on the class label, okay? In other words, conditioned on whether a cancer is malignant or benign, it models probability distribution over what the features of the cancer looks like.

Then having built this model – having built a model for PFX given Y and PFY, then by Bayes rule, obviously, you can compute PFY given 1, conditioned on X. This is just PFX given Y = 1 × PFX ÷ PFX, and, if necessary, you can calculate the denominator using this, right? And so by modeling PFX given Y and modeling PFY, you can actually use Bayes rule to get back to PFY given X, but a generative model – generative learning algorithm starts in modeling PFX given Y, rather than PFY given X, okay?

We'll talk about some of the tradeoffs, and why this may be a better or worse idea than a discriminative model a bit later. Let's go for a specific example of a generative learning algorithm, and for this specific motivating example, I'm going to assume that your input feature is X and RN and are continuous values, okay?

And under this assumption, let me describe to you a specific algorithm called Gaussian Discriminant Analysis, and the, I guess, core assumption is that we're going to assume in the Gaussian discriminant analysis model of that PFX given Y is Gaussian, okay?

So actually just raise your hand, how many of you have seen a multivariate Gaussian before – not a 1D Gaussian, but the higher range though? Okay, cool, like maybe half of you, two-thirds of you. So let me just say a few words about Gaussians, and for those of you that have seen it before, it'll be a refresher.

So we say that a random variable Z is distributed Gaussian, multivariate Gaussian as – and the script N for normal with parameters mean U and covariance sigma squared. If Z has a density 1 over 2 Pi, sigma 2, okay? That's the formula for the density as a generalization of the one dimension of Gaussians and no more the familiar bell-shape curve. It's a high dimension vector value random variable Z.

Don't worry too much about this formula for the density. You rarely end up needing to use it, but the two key quantities are this vector mew is the mean of the Gaussian and this matrix sigma is the covariance matrix – covariance, and so sigma will be equal to, right, the definition of covariance of a vector valued random variable is X - U, X - V conspose, okay?

And, actually, if this doesn't look familiar to you, you might re-watch the discussion section that the TAs held last Friday or the one that they'll be holding later this week on, sort of, a recap of probability, okay?

So multi-grade Gaussians is parameterized by a mean and a covariance, and let me just – can I have the laptop displayed, please? I'll just go ahead and actually show you, you know, graphically, the effects of varying a Gaussian – varying the parameters of a Gaussian. So what I have up here is the density of a zero mean Gaussian with covariance matrix equals the identity. The covariance matrix is shown in the upper right-hand corner of the slide, and there's the familiar bell-shaped curve in two dimensions.

And so if I shrink the covariance matrix, instead of covariance your identity, if I shrink the covariance matrix, then the Gaussian becomes more peaked, and if I widen the covariance, so like same = 2, 2, then the distribution – well, the density becomes more spread out, okay?

Those vectors stand at normal, identity covariance one. If I increase the diagonals of a covariance matrix, right, if I make the variables correlated, and the Gaussian becomes flattened out in this X = Y direction, and increase it even further, then my variables, X and Y, right – excuse me, it goes Z1 and Z2 are my two variables on a horizontal axis become even more correlated.

I'll just show the same thing in contours. The standard normal of distribution has contours that are – they're actually circles. Because of the aspect ratio, these look like ellipses. These should actually be circles, and if you increase the off diagonals of the Gaussian covariance matrix, then it becomes ellipses aligned along the, sort of, 45 degree angle in this example.

This is the same thing. Here's an example of a Gaussian density with negative covariances. So now the correlation goes the other way, so that even strong [inaudible] of covariance and the same thing in contours. This is a Gaussian with negative entries on the diagonals and even larger entries on the diagonals, okay?

And other parameter for the Gaussian is the mean parameters, so if this is – with mew0, and as he changed the mean parameter, this is mew equals 0.15, the location of the Gaussian just moves around, okay?

All right. So that was a quick primer on what Gaussians look like, and here's as a roadmap or as a picture to keep in mind, when we described the Gaussian discriminant analysis algorithm, this is what we're going to do. Here's the training set, and in the Gaussian discriminant analysis algorithm, what I'm going to do is I'm going to look at the positive examples, say the crosses, and just looking at only the positive examples, I'm gonna fit a Gaussian distribution to the positive examples, and so maybe I end up with a Gaussian distribution like that, okay? So there's PFX given Y = 1.

And then I'll look at the negative examples, the O's in this figure, and I'll fit a Gaussian to that, and maybe I get a Gaussian centered over there. This is the concept of my second Gaussian, and together – we'll say how later – together these two Gaussian densities will define a separator for these two classes, okay?

And it'll turn out that the separator will turn out to be a little bit different from what logistic regression gives you. If you run logistic regression, you actually get the division bound to be shown in the green line, whereas Gaussian discriminant analysis gives you the blue line, okay?

Switch back to chalkboard, please. All right. Here's the Gaussian discriminant analysis model, put into model PFY as a Bernoulli random variable as usual, but as a Bernoulli random variable and parameterized by parameter phi; you've seen this before. Model PFX given Y = 0 as a Gaussian – oh, you know what?

Yeah, yes, excuse me. I thought this looked strange. This should be a sigma, determined in a sigma to the one-half of the denominator there. It's no big deal. It was – yeah, well, okay. Right. I was listing the sigma to the determining the sigma to the one-half on a previous board, excuse me.

Okay, and so I model PFX given Y = 0 as a Gaussian with mean mew0 and covariance sigma to the sigma to the minus one-half, and – okay? And so the parameters of this model are phi, mew0, mew1, and sigma, and so I can now write down the likelihood of the parameters as – oh, excuse me, actually, the log likelihood of the parameters as the log of that, right?

So, in other words, if I'm given the training set, then they can write down the log likelihood of the parameters as the log of, you know, the probative probabilities of PFXI, YI, right? And this is just equal to that where each of these terms, PFXI given YI, or PFYI is then given by one of these three equations on top, okay?

And I just want to contrast this again with discriminative learning algorithms, right? So to give this a name, I guess, this sometimes is actually called the Joint Data Likelihood – the Joint Likelihood, and let me just contrast this with what we had previously when we're

talking about logistic regression. Where I said with the log likelihood of the parameter's theater was log of a product I = 1 to M, PFYI given XI and parameterized by a theater, right?

So back where we're fitting logistic regression models or generalized learning models, we're always modeling PFYI given XI and parameterized by a theater, and that was the conditional likelihood, okay, in which we're modeling PFYI given XI, whereas, now, regenerative learning algorithms, we're going to look at the joint likelihood which is PFXI, YI, okay?

So let's see. So given the training sets and using the Gaussian discriminant analysis model to fit the parameters of the model, we'll do maximize likelihood estimation as usual, and so you maximize your L with respect to the parameters phi, mew0, mew1, sigma, and so if we find the maximum likelihood estimate of parameters, you find that phi is – the maximum likelihood estimate is actually no surprise, and I'm writing this down mainly as a practice for indicating notation, all right?

So the maximum likelihood estimate for phi would be Sum over I, YI ÷ M, or written alternatively as Sum over – all your training examples of indicator YI = 1 ÷ M, okay? In other words, maximum likelihood estimate for a newly parameter phi is just the faction of training examples with label one, with Y equals 1. Maximum likelihood estimate for mew0 is this, okay? You should stare at this for a second and see if it makes sense.

Actually, I'll just write on the next one for mew1 while you do that. Okay? So what this is is what the denominator is sum of your training sets indicated YI = 0. So for every training example for which YI = 0, this will increment the count by one, all right?

So the denominator is just the number of examples with label zero, all right? And then the numerator will be, let's see, Sum from I = 1 for M, or every time YI is equal to 0, this will be a one, and otherwise, this thing will be zero, and so this indicator function means that you're including only the times for which YI is equal to one – only the turns which Y is equal to zero because for all the times where YI is equal to one, this sum and will be equal to zero, and then you multiply that by XI, and so the numerator is really the sum of XI's corresponding to examples where the class labels were zero, okay? Raise your hand if this makes sense. Okay, cool.

So just to say this fancifully, this just means look for your training set, find all the examples for which Y = 0, and take the average of the value of X for all your examples which Y = 0. So take all your negative fitting examples and average the values for X and that's mew0, okay?

If this notation is still a little bit cryptic – if you're still not sure why this equation translates into what I just said, do go home and stare at it for a while until it just makes sense. This is, sort of, no surprise. It just says to estimate the mean for the negative examples, take all your negative examples, and average them. So no surprise, but this is a useful practice to indicate a notation.

[Inaudible] divide the maximum likelihood estimate for sigma. I won't do that. You can read that in the notes yourself. And so having fit the parameters find mew0, mew1, and sigma to your data, well, you now need to make a prediction. You know, when you're given a new value of X, when you're given a new cancer, you need to predict whether it's malignant or benign.

Your prediction is then going to be, let's say, the most likely value of Y given X. I should write semicolon the parameters there. I'll just give that – which is the [inaudible] of a Y by Bayes rule, all right? And that is, in turn, just that because the denominator PFX doesn't depend on Y, and if PFY is uniform.

In other words, if each of your constants is equally likely, so if PFY takes the same value for all values of Y, then this is just arc X over Y, PFX given Y, okay?

This happens sometimes, maybe not very often, so usually you end up using this formula where you compute PFX given Y and PFY using your model, okay?

Student: Can you give us arc x?

Instructor (Andrew Ng):Oh, let's see. So if you take – actually let me. So the min of – arcomatics means the value for Y that maximizes this.

Student: Oh, okay.

Instructor (Andrew Ng):So just for an example, the min of X - 5 squared is 0 because by choosing X equals 5, you can get this to be zero, and the argument over X of X - 5 squared is equal to 5 because 5 is the value of X that makes this minimize, okay? Cool. Thanks for asking that.

Instructor (Andrew Ng):Okay. Actually any other questions about this? Yeah?

Student: Why is distributive removing? Why isn't [inaudible] –

Instructor (Andrew Ng):Oh, I see. By uniform I meant – I was being loose here. I meant if PFY = 0 is equal to PFY = 1, or if Y is the uniform distribution over the set 0 and 1.

Student: Oh.

Instructor (Andrew Ng):I just meant – yeah, if PFY = 0 zero = PFY given 1. That's all I mean, see? Anything else?

All right. Okay. So it turns out Gaussian discriminant analysis has an interesting relationship to logistic regression. Let me illustrate that. So let's say you have a training set – actually let me just go ahead and draw 1D training set, and that will kind of work, yes, okay.

So let's say we have a training set comprising a few negative and a few positive examples, and let's say I run Gaussian discriminate analysis. So I'll fit Gaussians to each of these two densities – a Gaussian density to each of these two – to my positive and negative training examples, and so maybe my positive examples, the X's, are fit with a Gaussian like this, and my negative examples I will fit, and you have a Gaussian that looks like that, okay?

Now, I hope this [inaudible]. Now, let's vary along the X axis, and what I want to do is I'll overlay on top of this plot. I'm going to plot PFY = 1 – no, actually, given X for a variety of values X, okay? So I actually realize what I should have done. I'm gonna call the X's the negative examples, and I'm gonna call the O's the positive examples. It just makes this part come in better.

So let's take a value of X that's fairly small. Let's say X is this value here on a horizontal axis. Then what's the probability of Y being equal to one conditioned on X? Well, the way you calculate that is you write PFY = 1 given X, and then you plug in all these formulas as usual, right? It's PFX given Y = 1, which is your Gaussian density, times PFY = 1, you know, which is essentially – this is just going to be equal to phi, and then divided by, right, PFX, and then this shows you how you can calculate this.

By using these two Gaussians and my phi on PFY, I actually compute what PFY = 1 given X is, and in this case, if X is this small, clearly it belongs to the left Gaussian. It's very unlikely to belong to a positive class, and so it'll be very small; it'll be very close to zero say, okay? And then we can increment the value of X a bit, and study a different value of X, and plot what is the PFY given X – PFY = 1 given X, and, again, it'll be pretty small.

Let's use a point like that, right? At this point, the two Gaussian densities have equal value, and if I ask if X is this value, right, shown by the arrow, what's the probably of Y being equal to one for that value of X? Well, you really can't tell, so maybe it's about 0.5, okay?

And if you fill in a bunch more points, you get a curve like that, and then you can keep going. Let's say for a point like that, you can ask what's the probability of X being one? Well, if it's that far out, then clearly, it belongs to this rightmost Gaussian, and so the probability of Y being a one would be very high; it would be almost one, okay?

And so you can repeat this exercise for a bunch of points. All right, compute PFY equals one given X for a bunch of points, and if you connect up these points, you find that the curve you get [Pause] plotted takes a form of sigmoid function, okay?

So, in other words, when you make the assumptions under the Gaussian discriminant analysis model, that PFX given Y is Gaussian, when you go back and compute what PFY given X is, you actually get back exactly the same sigmoid function that we're using which is the progression, okay?

But it turns out the key difference is that Gaussian discriminant analysis will end up choosing a different position and a steepness of the sigmoid than would logistic regression. Is there a question?

Student:I'm just wondering, the Gaussian of PFY [inaudible] you do?

Instructor (Andrew Ng):No, let's see. The Gaussian – so this Gaussian is PFX given Y = 1, and this Gaussian is PFX given Y = 0; does that make sense? Anything else?

Student: Okay.

Instructor (Andrew Ng):Yeah?

Student: When you drawing all the dots, how did you decide what Y given PFX was?

Instructor (Andrew Ng):What – say that again.

Student:I'm sorry. Could you go over how you figured out where to draw each dot?

Instructor (Andrew Ng):Let's see, okay. So the computation is as follows, right? The steps are I have the training sets, and so given my training set, I'm going to fit a Gaussian discriminant analysis model to it, and what that means is I'll build a model for PFX given Y = 1. I'll build a model for PFX given Y = 0, and I'll also fit a Bernoulli distribution to PFY, okay?

So, in other words, given my training set, I'll fit PFX given Y and PFY to my data, and now I've chosen my parameters of find mew0, mew1, and the sigma, okay? Then this is the process I went through to plot all these dots, right? It's just I pick a point in the X axis, and then I compute PFY given X for that value of X, and PFY given 1 conditioned on X will be some value between zero and one. It'll be some real number, and whatever that real number is, I then plot it on the vertical axis, okay?

And the way I compute PFY = 1 conditioned on X is I would use these quantities. I would use PFX given Y and PFY, and, sort of, plug them into Bayes rule, and that allows me to compute PFY given X from these three quantities; does that make sense?

Student: Yeah.

Instructor (Andrew Ng):Was there something more that –

Student: And how did you model PFX; is that –

Instructor (Andrew Ng):Oh, okay. Yeah, so – well, got this right here. So PFX can be written as, right, so PFX given Y = 0 × PFY = 0 + PFX given Y = 1, PFY = 1, right? And so each of these terms, PFX given Y and PFY, these are terms I can get out of, directly, from my Gaussian discriminant analysis model. Each of these terms is something that my

model gives me directly, so plugged in as the denominator, and by doing that, that's how I compute PFY = 1 given X, make sense?

Student: Thank you.

Instructor (Andrew Ng):Okay. Cool. So let's talk a little bit about the advantages and disadvantages of using a generative learning algorithm, okay? So in the particular case of Gaussian discriminant analysis, we assume that X conditions on Y is Gaussian, and the argument I showed on the previous chalkboard, I didn't prove it formally, but you can actually go back and prove it yourself is that if you assume X given Y is Gaussian, then that implies that when you plot Y given X, you find that – well, let me just write logistic posterior, okay?

And the argument I showed just now, which I didn't prove; you can go home and prove it yourself, is that if you assume X given Y is Gaussian, then that implies that the posterior distribution or the form of PFY = 1 given X is going to be a logistic function, and it turns out this implication in the opposite direction does not hold true, okay?

In particular, it actually turns out – this is actually, kind of, cool. It turns out that if you're seeing that X given Y = 1 is Hessian with parameter lambda 1, and X given Y = 0, is Hessian with parameter lambda 0. It turns out if you assumed this, then that also implies that PFY given X is logistic, okay?

So there are lots of assumptions on X given Y that will lead to PFY given X being logistic, and, therefore, this, the assumption that X given Y being Gaussian is the stronger assumption than the assumption that Y given X is logistic, okay? Because this implies this, right? That means that this is a stronger assumption than this because this, the logistic posterior holds whenever X given Y is Gaussian but not vice versa.

And so this leaves some of the tradeoffs between Gaussian discriminant analysis and logistic regression, right? Gaussian discriminant analysis makes a much stronger assumption that X given Y is Gaussian, and so when this assumption is true, when this assumption approximately holds, if you plot the data, and if X given Y is, indeed, approximately Gaussian, then if you make this assumption, explicit to the algorithm, then the algorithm will do better because it's as if the algorithm is making use of more information about the data. The algorithm knows that the data is Gaussian, right? And so if the Gaussian assumption, you know, holds or roughly holds, then Gaussian discriminant analysis may do better than logistic regression.

If, conversely, if you're actually not sure what X given Y is, then logistic regression, the discriminant algorithm may do better, and, in particular, use logistic regression, and maybe you see [inaudible] before the data was Gaussian, but it turns out the data was actually Poisson, right? Then logistic regression will still do perfectly fine because if the data were actually Poisson, then PFY = 1 given X will be logistic, and it'll do perfectly fine, but if you assumed it was Gaussian, then the algorithm may go off and do something that's not as good, okay?

So it turns out that – right. So it's slightly different. It turns out the real advantage of generative learning algorithms is often that it requires less data, and, in particular, data is never really exactly Gaussian, right? Because data is often approximately Gaussian; it's never exactly Gaussian.

And it turns out, generative learning algorithms often do surprisingly well even when these modeling assumptions are not met, but one other tradeoff is that by making stronger assumptions about the data, Gaussian discriminant analysis often needs less data in order to fit, like, an okay model, even if there's less training data.

Whereas, in contrast, logistic regression by making less assumption is more robust to your modeling assumptions because you're making a weaker assumption; you're making less assumptions, but sometimes it takes a slightly larger training set to fit than Gaussian discriminant analysis. Question?

Student: In order to meet any assumption about the number [inaudible], plus here we assume that PFY = 1, equal two number of. [Inaudible]. Is true when the number of samples is marginal?

Instructor (Andrew Ng):Okay. So let's see. So there's a question of is this true – what was that? Let me translate that differently. So the marving assumptions are made independently of the size of your training set, right? So, like, in least/great regression – well, in all of these models I'm assuming that these are random variables flowing from some distribution, and then, finally, I'm giving a single training set and that as for the parameters of the distribution, right?

Student: So what's the probability of Y = 1?

Instructor (Andrew Ng):Probability of Y + 1?

Student: Yeah, you used the –

Instructor (Andrew Ng):Sort of, this like – back to the philosophy of mass molecular estimation, right? I'm assuming that they're PFY is equal to phi to the Y, Y - phi to the Y or Y - Y. So I'm assuming that there's some true value of Y generating all my data, and then – well, when I write this, I guess, maybe what I should write isn't – so when I write this, I guess there are already two values of phi. One is there's a true underlying value of phi that guards the use to generate the data, and then there's the maximum likelihood estimate of the value of phi, and so when I was writing those formulas earlier, those formulas are writing for phi, and mew0, and mew1 were really the maximum likelihood estimates for phi, mew0, and mew1, and that's different from the true underlying values of phi, mew0, and mew1, but –

Student:[Off mic].

Instructor (Andrew Ng):Yeah, right. So maximum likelihood estimate comes from the data, and there's some, sort of, true underlying value of phi that I'm trying to estimate, and my maximum likelihood estimate is my attempt to estimate the true value, but, you know, by notational and convention often are just right as that as well without bothering to distinguish between the maximum likelihood value and the true underlying value that I'm assuming is out there, and that I'm only hoping to estimate.

Actually, yeah, so for the sample of questions like these about maximum likelihood and so on, I hope to tease to the Friday discussion section as a good time to ask questions about, sort of, probabilistic definitions like these as well. Are there any other questions? No, great. Okay.

So, great. Oh, it turns out, just to mention one more thing that's, kind of, cool. I said that if X given Y is Poisson, and you also go logistic posterior, it actually turns out there's a more general version of this. If you assume X given Y = 1 is exponential family with parameter A to 1, and then you assume X given Y = 0 is exponential family with parameter A to 0, then this implies that PFY = 1 given X is also logistic, okay? And that's, kind of, cool. It means that Y given X could be – I don't know, some strange thing. It could be gamma because we've seen Gaussian right next to the – I don't know, gamma exponential. They're actually a beta.

I'm just rattling off my mental list of exponential family extrusions.

It could be any one of those things, so [inaudible] the same exponential family distribution for the two classes with different natural parameters than the posterior PFY given 1 given X – PFY = 1 given X would be logistic, and so this shows the robustness of logistic regression to the choice of modeling assumptions because it could be that the data was actually, you know, gamma distributed, and just still turns out to be logistic. So it's the robustness of logistic regression to modeling assumptions.

And this is the density. I think, early on I promised two justifications for where I pulled the logistic function out of the hat, right? So one was the exponential family derivation we went through last time, and this is, sort of, the second one. That all of these modeling assumptions also lead to the logistic function. Yeah?

Student:[Off mic].

Instructor (Andrew Ng):Oh, that Y = 1 given as the logistic then this implies that, no. This is also not true, right? Yeah, so this exponential family distribution implies Y = 1 is logistic, but the reverse assumption is also not true. There are actually all sorts of really bizarre distributions for X that would give rise to logistic function, okay?

Okay. So let's talk about – those are first generative learning algorithm. Maybe I'll talk about the second generative learning algorithm, and the motivating example, actually this is called a Naive Bayes algorithm, and the motivating example that I'm gonna use will be spam classification.

All right. So let's say that you want to build a spam classifier to take your incoming stream of email and decide if it's spam or not. So let's see. Y will be 0 or 1, with 1 being spam email and 0 being non-spam, and the first decision we need to make is, given a piece of email, how do you represent a piece of email using a feature vector X, right? So email is just a piece of text, right? Email is like a list of words or a list of ASCII characters.

So I can represent email as a feature of vector X. So we'll use a couple of different representations, but the one I'll use today is we will construct the vector X as follows. I'm gonna go through my dictionary, and, sort of, make a listing of all the words in my dictionary, okay?

So the first word is RA. The second word in my dictionary is Aardvark, ausworth, okay? You know, and somewhere along the way you see the word "buy" in the spam email telling you to buy stuff. Tell you how you collect your list of words, you know, you won't find CS229, right, course number in a dictionary, but if you collect a list of words via other emails you've gotten, you have this list somewhere as well, and then the last word in my dictionary was zicmergue, which pertains to the technological chemistry that deals with the fermentation process in brewing.

So say I get a piece of email, and what I'll do is I'll then scan through this list of words, and wherever a certain word appears in my email, I'll put a 1 there. So if a particular email has the word "aid" then that's 1. You know, my email doesn't have the words ausworth or aardvark, so it gets zeros. And again, a piece of email, they want me to buy something, CS229 doesn't occur, and so on, okay? So this would be one way of creating a feature vector to represent a piece of email.

Now, let's throw the generative model out for this. Actually, let's use this. In other words, I want to model PFX given Y. The given Y = 0 or Y = 1, all right? And my feature vectors are going to be 0, 1 to the N. It's going to be these split vectors, binary value vectors. They're N dimensional. Where N may be on the order of, say, 50,000, if you have 50,000 words in your dictionary, which is not atypical. So values from – I don't know, mid-thousands to tens of thousands is very typical for problems like these.

And, therefore, there two to the 50,000 possible values for X, right? So two to 50,000 possible bit vectors of length 50,000, and so one way to model this is the multinomial distribution, but because there are two to the 50,000 possible values for X, I would need two to the 50,000, but maybe -1 parameters, right? Because you have this sum to 1, right? So -1. And this is clearly way too many parameters to model using the multinomial distribution over all two to 50,000 possibilities.

So in a Naive Bayes algorithm, we're going to make a very strong assumption on PFX given Y, and, in particular, I'm going to assume – let me just say what it's called; then I'll write out what it means. I'm going to assume that the XI's are conditionally independent given Y, okay?

Let me say what this means. So I have that PFX1, X2, up to X50,000, right, given the Y. By the key rule of probability, this is PFX1 given Y times PFX2 given Y, X1 times PF – I'll just put dot, dot, dot. I'll just write 1, 1 × dot, dot, dot up to, you know, well – whatever. You get the idea, up to PFX50,000, okay?

So this is the chain were of probability. This always holds. I've not made any assumption yet, and now, we're gonna meet what's called the Naive Bayes assumption, or this assumption that X defies a conditionally independent given Y. Going to assume that – well, nothing changes for the first term, but I'm gonna assume that PFX3 given Y, X1 is equal to PFX2 given the Y. I'm gonna assume that that term's equal to PFX3 given the Y, and so on, up to PFX50,000 given Y, okay? Or just written more compactly, means assume that PFX1, PFX50,000 given Y is the product from I = 1 to 50,000 or PFXI given the Y, okay?

And stating informally what this means is that I'm, sort of, assuming that – so unless you know the cost label Y, so long as you know whether this is spam or not spam, then knowing whether the word "A" appears in email does not affect the probability of whether the word "Ausworth" appears in the email, all right?

And, in other words, there's assuming – once you know whether an email is spam or not spam, then knowing whether other words appear in the email won't help you predict whether any other word appears in the email, okay? And, obviously, this assumption is false, right? This assumption can't possibly be true. I mean, if you see the word – I don't know, CS229 in an email, you're much more likely to see my name in the email, or the TA's names, or whatever.

So this assumption is normally just false under English, right, for normal written English, but it turns out that despite this assumption being, sort of, false in the literal sense, the Naive Bayes algorithm is, sort of, an extremely effective algorithm for classifying text documents into spam or not spam, for classifying your emails into different emails for your automatic view, for looking at web pages and classifying whether this webpage is trying to sell something or whatever. It turns out, this assumption works very well for classifying text documents and for other applications too that I'll talk a bit about later.

As a digression that'll make sense only to some of you. Let me just say that if you're familiar with Bayesian X world, say graphical models, the Bayesian network associated with this model looks like this, and you're assuming that this is random variable Y that then generates X1, X2, through X50,000, okay? If you've not seen the Bayes Net before, if you don't know your graphical model, just ignore this. It's not important to our purposes, but if you've seen it before, that's what it will look like.

Okay. So the parameters of the model are as follows with phi FI given Y = 1, which is probably FX = 1 or XI = 1 given Y = 1, phi I given Y = 0, and phi Y, okay? So these are the parameters of the model, and, therefore, to fit the parameters of the model, you can write down the joint likelihood, right, is equal to, as usual, okay?

So given the training sets, you can write down the joint likelihood of the parameters, and then when you do maximum likelihood estimation, you find that the maximum likelihood estimate of the parameters are – they're really, pretty much, what you'd expect. Maximum likelihood estimate for phi J given Y = 1 is sum from I = 1 to M, indicator XIJ = 1, YI = 1, okay?

And this is just a, I guess, stated more simply, the numerator just says, "Run for your entire training set, some [inaudible] examples, and count up the number of times you saw word "Jay" in a piece of email for which the label Y was equal to 1." So, in other words, look through all your spam emails and count the number of emails in which the word "Jay" appeared out of all your spam emails, and the denominator is, you know, sum from I = 1 to M, the number of spam. The denominator is just the number of spam emails you got.

And so this ratio is in all your spam emails in your training set, what fraction of these emails did the word "Jay" appear in – did the, "Jay" you wrote in your dictionary appear in? And that's the maximum likelihood estimate for the probability of seeing the word "Jay" conditions on the piece of email being spam, okay? And similar to your maximum likelihood estimate for phi Y is pretty much what you'd expect, right? Okay?

And so having estimated all these parameters, when you're given a new piece of email that you want to classify, you can then compute PFY given X using Bayes rule, right? Same as before because together these parameters gives you a model for PFX given Y and for PFY, and by using Bayes rule, given these two terms, you can compute PFX given Y, and there's your spam classifier, okay? Turns out we need one more elaboration to this idea, but let me check if there are questions about this so far.

Student: So does this model depend on the number of inputs?

Instructor (Andrew Ng):What do you mean, number of inputs, the number of features?

Student: No, number of samples.

Instructor (Andrew Ng):Well, N is the number of training examples, so this given M training examples, this is the formula for the maximum likelihood estimate of the parameters, right? So other questions, does it make sense? Or M is the number of training examples, so when you have M training examples, you plug them into this formula, and that's how you compute the maximum likelihood estimates.

Student: Is training examples you mean M is the number of emails?

Instructor (Andrew Ng):Yeah, right. So, right. So it's, kind of, your training set. I would go through all the email I've gotten in the last two months and label them as spam or not spam, and so you have – I don't know, like, a few hundred emails labeled as spam or not spam, and that will comprise your training sets for X1 and Y1 through XM, YM,

where X is one of those vectors representing which words appeared in the email and Y is 0, 1 depending on whether they equal spam or not spam, okay?

Student: So you are saying that this model depends on the number of examples, but the last model doesn't depend on the models, but your phi is the same for either one.

Instructor (Andrew Ng):They're different things, right? There's the model which is – the modeling assumptions aren't made very well. I'm assuming that – I'm making the Naive Bayes assumption. So the probabilistic model is an assumption on the joint distribution of X and Y. That's what the model is, and then I'm given a fixed number of training examples. I'm given M training examples, and then it's, like, after I'm given the training sets, I'll then go in to write the maximum likelihood estimate of the parameters, right? So that's, sort of, maybe we should take that offline for – yeah, ask a question?

Student: Then how would you do this, like, if this [inaudible] didn't work?

Instructor (Andrew Ng):Say that again.

Student: How would you do it, say, like the 50,000 words –

Instructor (Andrew Ng):Oh, okay. How to do this with the 50,000 words, yeah. So it turns out this is, sort of, a very practical question, really. How do I count this list of words? One common way to do this is to actually find some way to count a list of words, like go through all your emails, go through all the – in practice, one common way to count a list of words is to just take all the words that appear in your training set.

That's one fairly common way to do it, or if that turns out to be too many words, you can take all words that appear at least three times in your training set. So words that you didn't even see three times in the emails you got in the last two months, you discard. So those are – I was talking about going through a dictionary, which is a nice way of thinking about it, but in practice, you might go through your training set and then just take the union of all the words that appear in it.

In some of the tests I've even, by the way, said select these features, but this is one way to think about creating your feature vector, right, as zero and one values, okay? Moving on, yeah. Okay. Ask a question?

Student:I'm getting, kind of, confused on how you compute all those parameters.

Instructor (Andrew Ng):On how I came up with the parameters?

Student: Correct.

Instructor (Andrew Ng):Let's see. So in Naive Bayes, what I need to do – the question was how did I come up with the parameters, right? In Naive Bayes, I need to build a model for PFX given Y and for PFY, right? So this is, I mean, in generous of learning

algorithms, I need to come up with models for these. So how'd I model PFY? Well, I just those to model it using a Bernoulli distribution, and so PFY will be parameterized by that, all right?

Student: Okay.

Instructor (Andrew Ng):And then how'd I model PFX given Y? Well, let's keep changing bullets. My model for PFX given Y under the Naive Bayes assumption, I assume that PFX given Y is the product of these probabilities, and so I'm going to need parameters to tell me what's the probability of each word occurring, you know, of each word occurring or not occurring, conditions on the email being spam or not spam email, okay?

Student: How is that Bernoulli?

Instructor (Andrew Ng):Oh, because X is either zero or one, right? By the way I defined the feature vectors, XI is either one or zero, depending on whether words I appear as in the email, right? So by the way I define the feature vectors, XI – the XI is always zero or one. So that by definition, if XI, you know, is either zero or one, then it has to be a Bernoulli distribution, right? If XI would continue as then you might model this as Gaussian and say you end up like we did in Gaussian discriminant analysis. It's just that the way I constructed my features for email, XI is always binary value, and so you end up with a Bernoulli here, okay? All right. I should move on.

So it turns out that this idea almost works. Now, here's the problem. So let's say you complete this class and you start to do, maybe do the class project, and you keep working on your class project for a bit, and it becomes really good, and you want to submit your class project to a conference, right? So, you know, around – I don't know, June every year is the conference deadline for the next conference. It's just the name of the conference; it's an acronym.

And so maybe you send your project partners or senior friends even, and say, "Hey, let's work on a project and submit it to the NIPS conference." And so you're getting these emails with the word "NIPS" in them, which you've probably never seen before, and so a piece of email comes from your project partner, and so you go, "Let's send a paper to the NIPS conference."

And then your stamp classifier will say PFX – let's say NIPS is the 30,000th word in your dictionary, okay? So X30,000 given the 1, given Y = 1 will be equal to 0. That's the maximum likelihood of this, right? Because you've never seen the word NIPS before in your training set, so maximum likelihood of the parameter is that probably have seen the word NIPS is zero, and, similarly, you know, in, I guess, non-spam mail, the chance of seeing the word NIPS is also estimated as zero.

So when your spam classifier goes to compute PFY = 1 given X, it will compute this right here × PFY over – well, all right. And so you look at that terms, say, this will be

product from I = 1 to 50,000, PFXI given Y, and one of those probabilities will be equal to zero because PFX30,000 = 1 given Y = 1 is equal to zero. So you have a zero in this product, and so the numerator is zero, and in the same way, it turns out the denominator will also be zero, and so you end up with – actually all of these terms end up being zero. So you end up with PFY = 1 given X is 0 over 0 + 0, okay, which is undefined.

And the problem with this is that it's just statistically a bad idea to say that PFX30,000 given Y is 0, right? Just because you haven't seen the word NIPS in your last two months worth of email, it's also statistically not sound to say that, therefore, the chance of ever seeing this word is zero, right?

And so is this idea that just because you haven't seen something before, that may mean that that event is unlikely, but it doesn't mean that it's impossible, and just saying that if you've never seen the word NIPS before, then it is impossible to ever see the word NIPS in future emails; the chance of that is just zero.

So we're gonna fix this, and to motivate the fix I'll talk about – the example we're gonna use is let's say that you've been following the Stanford basketball team for all of their away games, and been, sort of, tracking their wins and losses to gather statistics, and, maybe – I don't know, form a betting pool about whether they're likely to win or lose the next game, okay?

So these are some of the statistics. So on, I guess, the 8th of February last season they played Washington State, and they did not win. On the 11th of February, they play Washington, 22nd they played USC, played UCLA, played USC again, and now you want to estimate what's the chance that they'll win or lose against Louisville, right?

So find the four guys last year or five times and they weren't good in their away games, but it seems awfully harsh to say that – so it seems awfully harsh to say there's zero chance that they'll win in the last – in the 5th game. So here's the idea behind Laplace smoothing which is that we're estimate the probably of Y being equal to one, right? Normally, the maximum likelihood [inaudible] is the number of ones divided by the number of zeros plus the number of ones, okay?

I hope this informal notation makes sense, right? Knowing the maximum likelihood estimate for, sort of, a win or loss for Bernoulli random variable is just the number of ones you saw divided by the total number of examples. So it's the number of zeros you saw plus the number of ones you saw.

So in the Laplace Smoothing we're going to just take each of these terms, the number of ones and, sort of, add one to that, the number of zeros and add one to that, the number of ones and add one to that, and so in our example, instead of estimating the probability of winning the next game to be 0 ÷ 5 + 0, we'll add one to all of these counts, and so we say that the chance of their winning the next game is 1/7th, okay? Which is that having seen them lose, you know, five away games in a row, we aren't terribly – we don't think it's terribly likely they'll win the next game, but at least we're not saying it's impossible.

As a historical side note, the Laplace actually came up with the method. It's called the Laplace smoothing after him. When he was trying to estimate the probability that the sun will rise tomorrow, and his rationale was in a lot of days now, we've seen the sun rise, but that doesn't mean we can be absolutely certain the sun will rise tomorrow. He was using this to estimate the probability that the sun will rise tomorrow. This is, kind of, cool.

So, and more generally, if Y takes on K possible of values, if you're trying to estimate the parameter of the multinomial, then you estimate PFY = 1. Let's see. So the maximum likelihood estimate will be Sum from J = 1 to M, indicator YI = J ÷ M, right? That's the maximum likelihood estimate of a multinomial probability of Y being equal to – oh, excuse me, Y = J. All right. That's the maximum likelihood estimate for the probability of Y = J, and so when you apply Laplace smoothing to that, you add one to the numerator, and add K to the denominator, if Y can take up K possible values, okay?

So for Naive Bayes, what that gives us is – shoot. Right? So that was the maximum likelihood estimate, and what you end up doing is adding one to the numerator and adding two to the denominator, and this solves the problem of the zero probabilities, and when your friend sends you email about the NIPS conference, your spam filter will still be able to make a meaningful prediction, all right? Okay. Shoot. Any questions about this? Yeah?

Student: So that's what doesn't makes sense because, for instance, if you take the games on the right, it's liberal assumptions that the probability of winning is very close to zero, so, I mean, the prediction should be equal to PF, 0.

Instructor (Andrew Ng):Right. I would say that in this case the prediction is 1/7th, right? We don't have a lot of – if you see somebody lose five games in a row, you may not have a lot of faith in them, but as an extreme example, suppose you saw them lose one game, right? It's just not reasonable to say that the chances of winning the next game is zero, but that's what maximum likelihood estimate will say.

Student: Yes.

Instructor (Andrew Ng):And –

Student: In such a case anywhere the learning algorithm [inaudible] or –

Instructor (Andrew Ng):So some questions of, you know, given just five training examples, what's a reasonable estimate for the chance of winning the next game, and 1/7th is, I think, is actually pretty reasonable. It's less than 1/5th for instance. We're saying the chances of winning the next game is less than 1/5th.

It turns out, under a certain set of assumptions I won't go into – under a certain set of Bayesian assumptions about the prior and posterior, this Laplace smoothing actually gives the optimal estimate, in a certain sense I won't go into of what's the chance of

winning the next game, and so under a certain assumption about the Bayesian prior on the parameter. So I don't know. It actually seems like a pretty reasonable assumption to me. Although, I should say, it actually turned out –

No, I'm just being mean. We actually are a pretty good basketball team, but I chose a losing streak because it's funnier that way. Let's see. Shoot. Does someone want to – are there other questions about this? No, yeah. Okay. So there's more that I want to say about Naive Bayes, but we'll do that in the next lecture. So let's wrap it for today.

[End of Audio]

Duration: 76 minutes