Lecture 15

Continuing with linear regression: we attempted to find the linear function h using least squares, which yielded:

$$\beta_{LS}(X^TX)^{-1}X^Ty \tag{1}$$

and we showed that you could derive this formula either by progressing with either the minimizational technique of the maximization technique. This is great, because it's a deterministic unbiased process.

Logistic Regression

So far, we have talked about y being a continuous variable. What happens when y is categorical, and not continuous? Well, here we are dealing with **Logistic Regression**, that is, we now have labeled data where each point y_i is labeled with a certain category.

Let's assume we have 2 classes. Even if we can make these classes numerical (e.g., translate yes/no to 1/0), these numbers don't have a mathematical meaning in the context of linear regression - these numbers don't represent a continuous distribution. For example, the fact that we translated yes/no to 1/0 is arbitrary - we could have done 10/20 instead, and now we might learn a completely different model in linear regression! Thus, we need some other kind of mathematical representation.

Alright, well, perhaps we can use the probability of belonging to a given class as a proxy for confidently we can classify a given point? Perhaps we can then fit a linear model to the probability of being in a given class! That is, maybe we can fit a linear model of a probability distribution, so that when we get a new record, we output a probability of belonging to a certain class.

How can we do this? If all values are either 0 or 1, then how can we ensure that β_{LS} is always between 0 and 1 for an unseen x? Well, we can't do this.

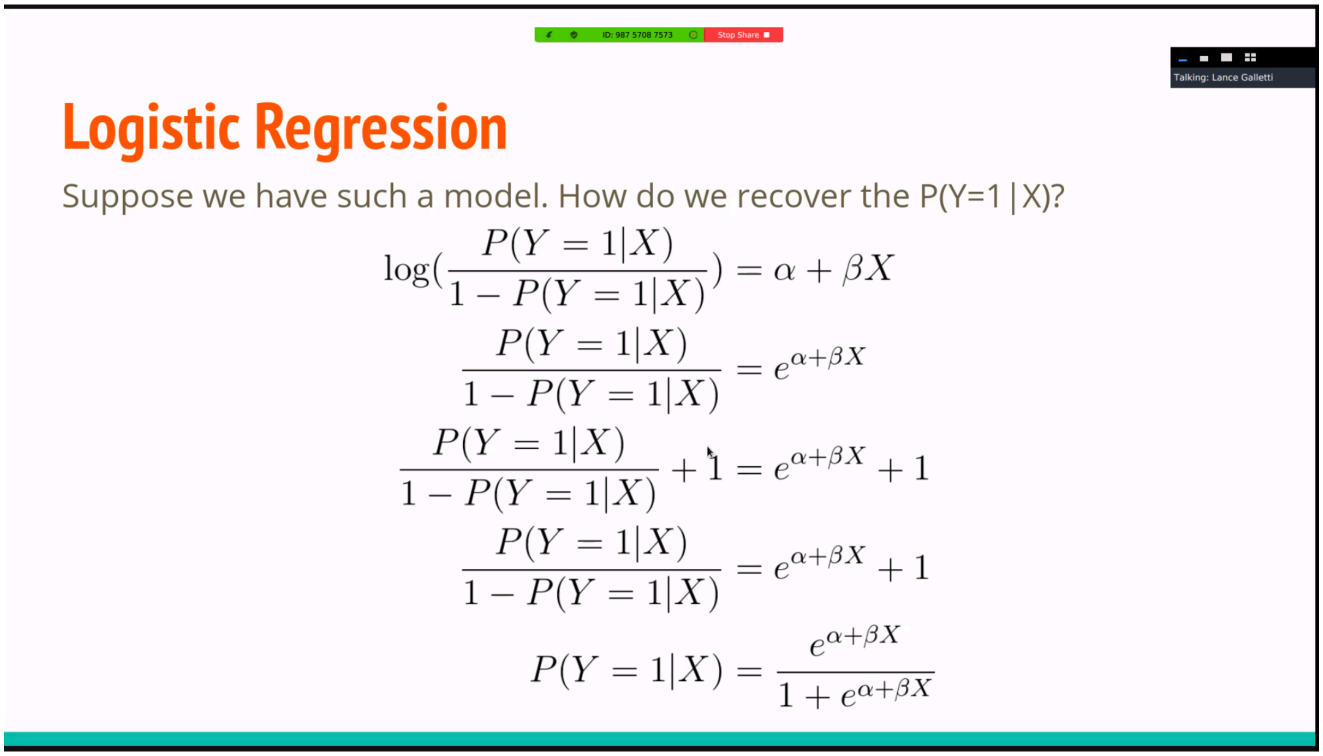
Instead, we can try to extent the possible range: that is, we define the odds as $P/(1-p)$, where $p = P(Y = \text{class } 1|X)$. Unfortunately, we can still find a new value x that falls outside our range of outputs.

We need to further extend our range to $(-\infty, \infty)$. How do we do this? We can do this by taking the log. This is also convenient numerically, because in the previous odds format, tiny variations in p have large effects on the odds. Now, our model is:

$$log(\frac{P(Y=1|X)}{1-P(Y=1|X)}) = \alpha + \beta X \tag{2}$$

Let's translate this equation: what is the log of the odds of being in class 1, and we're going to try and fit a linear model to that. This is important, because we're going to use this to motivate and understand neural networks.

Suppose we have such a model. How do we recover the probability that an instance belongs to class 1?



Logistic Regression

Suppose we have such a model. How do we recover the P(Y=1|X)?

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \alpha + \beta X$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} = e^{\alpha+\beta X}$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} + 1 = e^{\alpha+\beta X} + 1$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} = e^{\alpha+\beta X} + 1$$

$$P(Y=1|X) = \frac{e^{\alpha+\beta X}}{1+e^{\alpha+\beta X}}$$

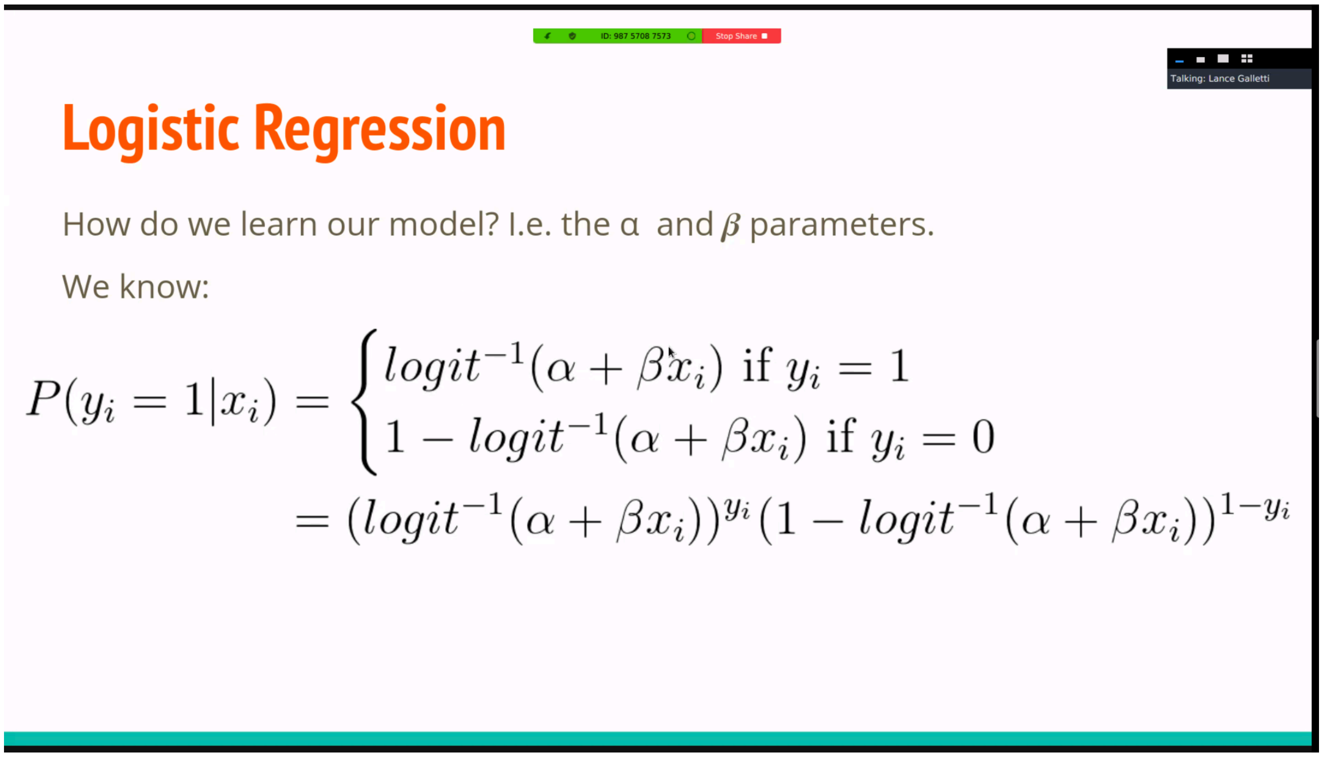
There's actually a typo here: in the second to bottom line, the numerator should just be 1. The final result is correct.

The funtion that we apply to the probability in order to obtain the log odds is called the **logit** function. The function used to retrieve our probability from the log odds is called the **logit inverse**

Now, we can say: if the probability is greater than 0.5, classify it as class 1, otherwise classify it as class 0.

Learning the Model

How do we find the parameters α and β? Consider what we know:



Logistic Regression

How do we learn our model? I.e. the α and β parameters.

We know:

$$P(y_i = 1|x_i) = \begin{cases} logit^{-1}(\alpha + \beta x_i) \text{ if } y_i = 1 \\ 1 - logit^{-1}(\alpha + \beta x_i) \text{ if } y_i = 0 \end{cases}$$
$$= (logit^{-1}(\alpha + \beta x_i))^{y_i}(1 - logit^{-1}(\alpha + \beta x_i))^{1-y_i}$$

Because we have two different cases, we can transform the two equations into one equation, by making y_i the exponent. We now have one single equation for understanding the probability.

We can now define a loss function, that tells us the probability of having seen the data that was actually observed. Assuming each was independent, it's just the multiplicative sum of each probability.

Then, given that loss function, we can try to maximize alpha and beta to maximize the probability that we saw. Unfortunately, this is not so simple, because we don't have a closed form solution to solve this. Instead, we have to use numerical approximation methods in order to solve this optimization problem. Nonetheless, the idea is similar to things we've seen before: start with random values for alpha and beta, and then attempt to update each of the values for an improvement. Eventually, we'll use gradient descent in order to recover these values.

Note: this is just the binary case, and we can extend this to multiple classes.

Let's make some notation clear:

- y_i is the "true" value from our data set. Read as "y-i"
- $\hat{y}_i$ is the estimate of y_i from our model. Read as "y-i-hat"
- $\bar{y}$ is the sample mean of all y_i. Read as "y-bar," the mean.
- $y_i - \hat{y}_i$ are the estimates ϵ_i, and are referred to as residuals. It is the noise present in the true relation between x and y.

We can evaluate our model with a couple different methods:

- TSS: total sum of squares: $\sum_i^n (y_i - \bar{y})^2$
- RSS: residual sum of squares: $\sum_i^n (y_i - \hat{y}_i)^2$
- ESS: explained sum of squares: $\sum_i^n (\hat{y}_i - \bar{y})^2$

Then, we have an R^2 value, which is $R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$, and the range is $(0, 1)$. Which is a good value for R^2? Is it better to be closer to 0 or 1? Well, if TSS and ESS are about the same, then that means that the variance explained by the model almost totally captures the variance seen in the data - that is a good thing. Thus, 1 is better, and 0 is worse.