STAT340 Lecture 10: logistic regression

Brian Powers

Updated 11/18/2024

Introduction

These notes concern the problem of *logistic regression*. This is a regression method that allows us to handle the situation where we have predictors and responses, but the predictors are binary rather than continuous. Fitting a linear regression model in this situation would be a bit silly, for reasons illustrated below. Logistic regression lets us keep all the “nice” things about linear regression (i.e., that we like linear functions), while being better suited to this different data setting.

Learning objectives

After this lecture, you will be able to

- Explain the motivation for logistic regression
- Fit a logistic regression model to data with one or more predictors and a binary response
- Interpret estimated coefficients in a logistic regression model

Logistic regression: motivation I

Consider the following data, which you'll play with more in discussion section: the Pima Indian data set.

has diabetes?

```
library(MASS)
head(Pima.te)
##   npreg glu bp skin  bmi   ped age type
## 1     6 148 72   35 33.6 0.627  50  Yes
## 2     1  85 66   29 26.6 0.351  31   No
## 3     1  89 66   23 28.1 0.167  21   No
## 4     3  78 50   32 31.0 0.248  26  Yes
## 5     2 197 70   45 30.5 0.158  53  Yes
## 6     5 166 72   19 25.8 0.587  51  Yes
```

This data was collection by the US National Institute of Diabetes and Digestive and Kidney Diseases. Each row of this data set corresponds to a woman of Pima Indian descent living near Phoenix Arizona. The columns include a number of biomarkers (e.g., glucose levels, blood pressure and age), as well as a column indicating whether or not each subject had diabetes (as measured according to measures laid out by the World Health Organization). See `?Pima.te` for more information.

Logistic regression: motivation II

Can we predict whether or not a given person has diabetes based on their biomarkers? It is natural to cast this as a prediction problem just like the linear regression problems we have discussed in recent weeks: we are given pairs (X_i, Y_i), where X_i is a vector of predictors and Y_i is a response. The difference is that now, Y_i is more naturally thought of as a binary label (in this case, indicating whether or not a subject has diabetes), instead of a continuous number.

Example: image classification

Suppose that we have a collection of images, and our goal is to detect whether or not a given image has a cat in it. Our data takes the form of a collection of pairs (X_i, Y_i), $i = 1, 2, \ldots, n$ where X_i is an image (e.g., a bunch of a pixels) and

$$Y_i = \begin{cases} 1 & \text{if image } i \text{ contains a cat} \\ 0 & \text{if image } i \text{ does not contain a cat.} \end{cases}$$

Caution: just because we encode yes as 1
NO as 0
can we treat the response as
a quantitative variable?

Example: fraud detection

An online banking or credit card service might like to be able to detect whether or not a given transaction is fraudulent. Predictors X_i might take the form of things like transaction amount, location and time, and the binary label Y_i corresponds to whether or not the transaction is fraudulent.

Example: brain imaging

In own work in neuroscience, a common task is to detect whether or not a subject has a disease or disorder (e.g., Parkinson's or schizophrenia) based on brain imaging data obtained via, for example, functional magnetic resonance imaging (fMRI). Our labels Y_i are given by

$$Y_i = \begin{cases} 1 & \text{if subject } i \text{ has the disease/disorder} \\ 0 & \text{if subject } i \text{ does not have the disease/disorder.} \end{cases}$$

Our predictors X_i for subject $i \in \{1, 2, \ldots, n\}$ might consist of a collection of features derived from the brain imagining (e.g., average pixel values or correlations between different locations in the brain). In my own research, X_i is a network that describes brain structure, constructed from the imaging data. In other studies, X_i might even be the brain imaging data itself. Our goal is to predict Y_i based on the observed imaging data (or image-derived features) X_i.

Formulating a model

A natural approach to this problem would be to just take our existing knowledge of linear regression and apply it here. If our predictor $\underline{X_i}$ has p dimensions,

$$X_i = (X_{i,1}, X_{i,2}, X_{i,3}, \ldots, X_{i,p})^T \in \mathbb{R}^p,$$

then we might try to fit a model of the form

$$\underline{Y_i} = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_p X_{i,p} \quad \boxed{+\text{ error }?}$$

The obvious problem with this is that the right-hand side of this equation is an arbitrary real number, whereas the left-hand side (i.e., Y_i) is 0 or 1.

Example: linear regression on the Pima data set I

For easier coding later, let's add a column to the Pima.te data set. The type column is Yes or No according to whether or not each subject has diabetes. Let's add a column with a more straightforward name and which directly encodes this diabetes status as 0 or 1.

```
Pima.te$diabetes <- ifelse( Pima.te$type=='Yes', 1, 0)
head(Pima.te)
##   npreg glu bp skin  bmi   ped age type diabetes
## 1     6 148 72   35 33.6 0.627  50  Yes        1
## 2     1  85 66   29 26.6 0.351  31   No        0
## 3     1  89 66   23 28.1 0.167  21   No        0
## 4     3  78 50   32 31.0 0.248  26  Yes        1
## 5     2 197 70   45 30.5 0.158  53  Yes        1
## 6     5 166 72   19 25.8 0.587  51  Yes        1
```

Example: linear regression on the Pima data set II

Now, let's pick a predictor and fit a simple linear regression model.

Note
This is not the right way to fit a 0/1 responce

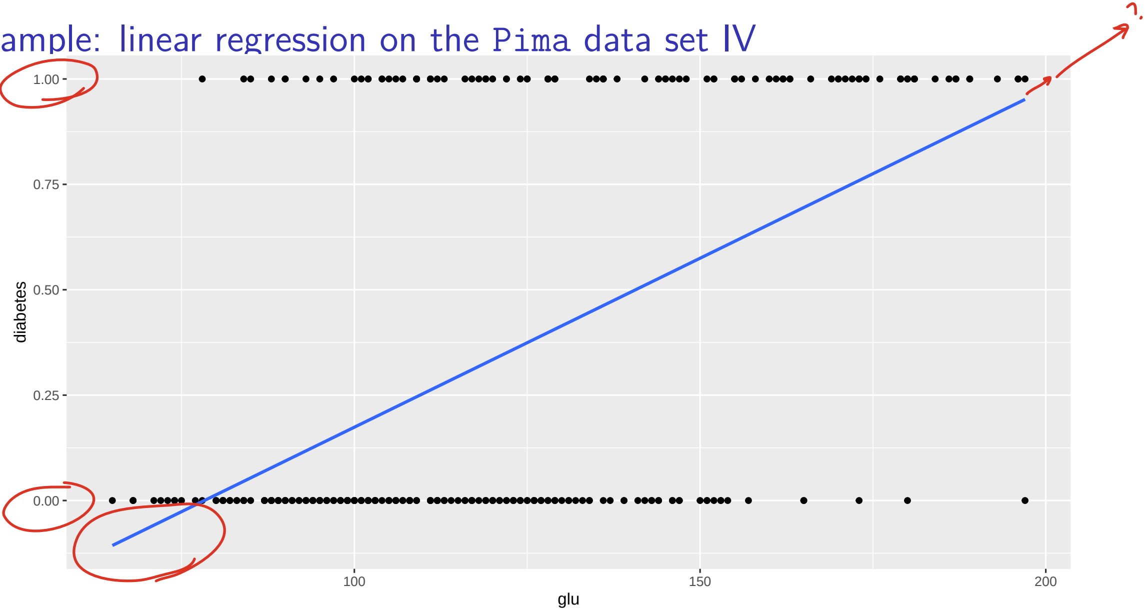
```
pima_lm <- lm( diabetes ~ 1 + glu, data=Pima.te );
summary(pima_lm)
##
## Call:
## lm(formula = diabetes ~ 1 + glu, data = Pima.te)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9516 -0.2701 -0.1138  0.2408  1.0025
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.6278036  0.0892443  -7.035 1.16e-11 ***
## glu          0.0080171  0.0007251  11.057  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4023 on 330 degrees of freedom
## Multiple R-squared:  0.2703, Adjusted R-squared:  0.2681
## F-statistic: 122.3 on 1 and 330 DF,  p-value: < 2.2e-16
```

Example: linear regression on the `Pima` data set III

Looks like we managed to capture some signal there, at least according to the F-statistic! Let's look at a plot.

```r
library(ggplot2)
pp <- ggplot(data=Pima.te, aes(x=glu, y=diabetes)) + geom_point()
pp <- pp + geom_smooth(method='lm', se=FALSE, formula='y ~ 1 + x')
```

Example: linear regression on the Pima data set IV



Example: linear regression on the `Pima` data set V

Hmm... so our fitted model takes in a number (i.e., `glu`) and outputs a number that is our prediction for Y_i. The trouble is that our responses are binary, but our outputs can be anything. If our goal is to predict y, and we predict y to be $\hat{\beta}_0 + \hat{\beta}_1 x$, then we can (potentially) predict arbitrary large or small values for y, if x is suitably large or small. That's a bit of an awkward fact– after all, we want our prediction to be 0 or 1. This isn't the end of the world. We could do something like "clipping" our output $\hat{\beta}_0 + \hat{\beta}_1 x$ to whichever of 0 or 1 is closest or something like that. But this feels clumsy, at best.

Example: linear regression on the `Pima` data set VI

Is there a more appropriate approach?

Well, there are many, but an especially simple one is *logistic regression*, which seeks to keep the “linear combination of predictors” idea that we like so much from linear regression, but modify how we make predictions to be better-suited to the “binary response” setting.

Logistic regression I

The core problem with applying linear regression to a binary response is that the output of linear regression can be any number, while we are really only interested in outputs in $\{0,1\}$. How might we transform our *linear* prediction, $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$, into something more sensible?

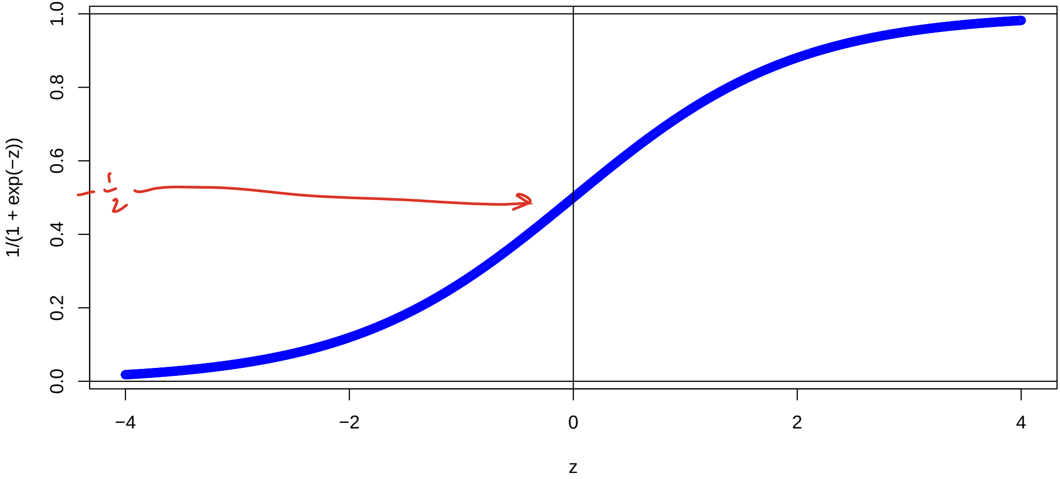
Logistic regression solves this problem by taking our linear prediction $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ and turning it into a *probability*, i.e., a number between 0 and 1. We do that using the *logistic function*,

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}.$$

Logistic regression II

Here is a plot of the function:

$$Y = \frac{1}{1+e^{-z}}$$



The logistic function is an example of a sigmoid function (fancy Greek for “S-shaped”).

Logistic regression III

Using the logistic function, we can modify the output of our model by passing our linear regression model's prediction, which is a real number, as the input of the logistic function so that it outputs a number between zero and one.

$\sigma(\hat{y}) \in (0,1)$ $\hat{y} \in (-\infty,\infty)$

$$\sigma\left(\hat{y}\right) = \sigma\left(\hat{\beta}_0 + \hat{\beta}_1 x\right) = \frac{1}{1 + \exp\{-(\hat{\beta}_0 + \hat{\beta}_1 x)\}} = \frac{\exp\{\hat{\beta}_0 + \hat{\beta}_1 x\}}{1 + \exp\{\hat{\beta}_0 + \hat{\beta}_1 x\}}$$

The especially nice thing about this is that since this number is between 0 and 1, we can interpret this as a probability:

$$\Pr[Y_i = 1; X_i = x, \beta_0, \beta_1] = \sigma\left(\beta_0 + \beta_1 x\right) = \frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}}$$

The semi-colon notation there is just to stress that we are treating β_0 and β_1 as parameters and the data x as an *input*, but *not* as things that we are conditioning on.

Logistic regression IV

Aside: there is a whole branch of statistics called *Bayesian statistics* that seeks to treat parameters like our coefficients β_0 and β_1 as variables that we can condition on. It's a very cool area, especially useful in machine learning, but it will have to wait for later courses!

Now, when it comes time to make a prediction on input x, we can pass $\beta_0 + \beta_1 x$ into the sigmoid function, and predict

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(\beta_0 + \beta_1 x) > \frac{1}{2} \\ 0 & \text{if } \sigma(\beta_0 + \beta_1 x) \leq \frac{1}{2}. \end{cases}$$

Logistic regression V

In the multivariate setting, where our i-th predictor X_i takes the form

$$X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p}),$$

multiple logistic regression is the straight-forward extension of this idea. Given (estimated) coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$, we output a probability

$$\begin{aligned}\Pr[Y_i = 1; X_i = (x_1, x_2, \ldots, x_p), \beta_0, \beta_1, \ldots, \beta_p] \\ &= \sigma\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p\right) \\ &= \frac{1}{1 + \exp\{-(\underline{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p})\}}\end{aligned}$$

Logistic regression in R I

Let's ignore, for now, the question of *how* we fit this model (spoiler alert: least squares isn't going to make a lot of sense anymore; we'll need something more clever), let's ask R to fit a logistic regression model to our problem of predicting diabetes from the `glu` variable in the Pima data set.

The function that we use in R is `glm`. "GLM" stands for *generalized* linear model. That is, we are generalizing linear regression. In particular, we are generalizing linear regression by doing linear regression, but then passing the linear regression prediction $\beta_0 + \beta_1 x$ through another function.

To perform *logistic* regression, we need to specify to R that we are using a "binomial" family of responses– our response Y is binary, so it can be thought of as a Binomial random variable, with $\Pr[Y = 1] = \sigma(\beta_0 + \beta_1 X)$.

Other than that, fitting a model with `glm` is basically the same as using plain old `lm`.

Logistic regression in R II

Even the model summary output looks about the same:

```
pima_logistic <- glm( diabetes ~ 1 + glu, data=Pima.te, family=binomial );
summary(pima_logistic)
##
## Call:
## glm(formula = diabetes ~ 1 + glu, family = binomial, data = Pima.te)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.946808   0.659839  -9.013   <2e-16 ***
## glu          0.042421   0.005165   8.213   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.30  on 331  degrees of freedom
## Residual deviance: 325.99  on 330  degrees of freedom
## AIC: 329.99
##
## Number of Fisher Scoring iterations: 4
```

no residuals

z not t

all new

- no R^2
- no F stat
- no RSE

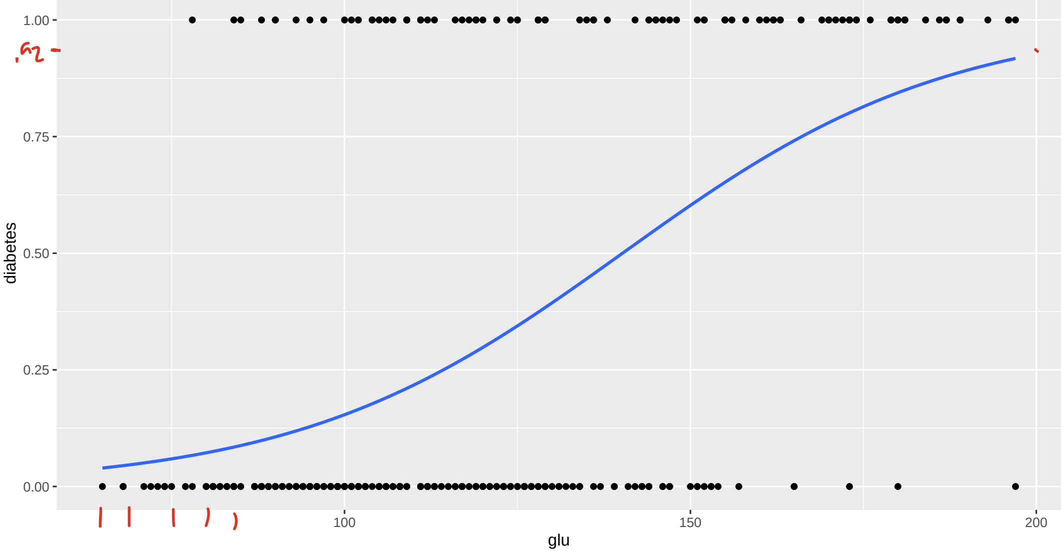
Logistic regression in R III

But notice now that our model's outputs are always between 0 and 1:

```r
pp <- ggplot( Pima.te, aes(x=glu, y=diabetes) ) + geom_point();
pp <- pp + geom_smooth(formula='y ~ 1+x', se=FALSE,
                       method='glm',
                       method.args=list(family = "binomial") );
# Note that geom_smooth needs an extra list() of arguments to specify the
# extra arguments that we passed to the glm() function above.
# In particular, we need to tell the GLM to use a binomial response.
pp
```

predictions

Logistic regression in R IV



Logistic regression in R V

If we interpret our model's output as a probability, namely the probability that we observe label Y for predictors X, then it is clear that as `glu` increases, our model thinks it is more likely that a person has diabetes.

Just as with linear regression, our logistic model object will make predictions for new, previously unseen predictors if we use the `predict` function.

```r
# Reminder: we pass a data frame into the predict function, and our model
# will produce an output for ease row of that data frame.
# So this is making a prediction for a subject with `glu=200`.
# Looking at our scatter plot above, it's clear that `glu=200` is at the
# far upper end of our data distribution, so we should expect our model to
# produce an output close to 1.
predict(pima_logistic, type='response', newdata = data.frame(glu=200) )
##         1
## 0.9267217
```

this optn. givs transformd 0/1 values

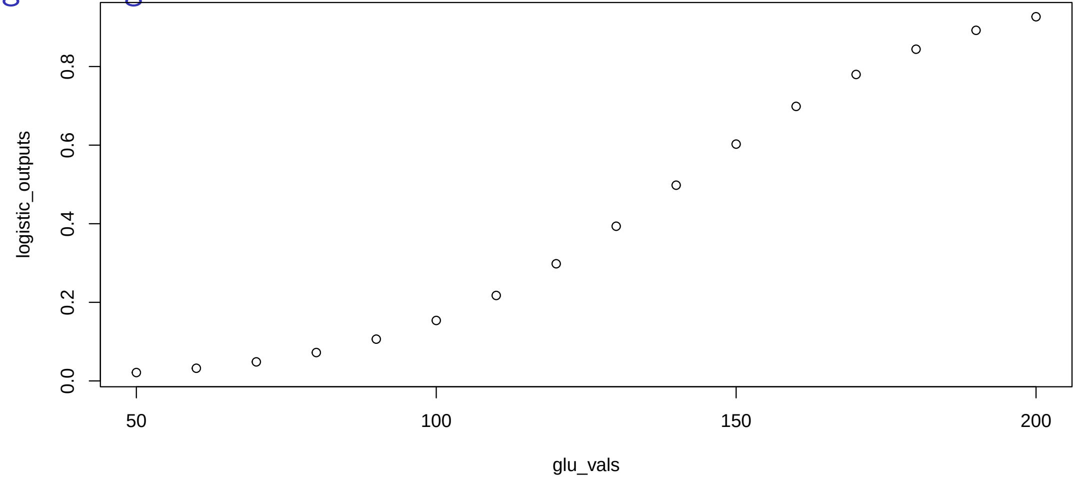
$$\hat{\Pr}[Y=1 \mid X=200] = .9267$$

Logistic regression in R VI

Generally speaking, as `glu` increases, our model is more confident that the subject has diabetes.

```r
glu_vals <- seq(50,200,10);
logistic_outputs <- predict(pima_logistic, type='response',
                            newdata = data.frame(glu=glu_vals) )
plot( glu_vals, logistic_outputs)
```

Logistic regression in R VII



Logistic regression in R VIII

Again, these are our model's predicted outputs for given input values for glucose levels. As `glu` increases, our model assigns higher and higher probabilities to the possibility that a subject has diabetes.

You'll play around with logistic regression and the Pima data set more in discussion section, where you'll find that if we choose our predictors more carefully (and use more predictors), we can build a pretty good model!

Interpreting model coefficients

In linear regression, our model coefficients had a simple interpretation: β_1 was the change in response associated with a unit change in the corresponding predictor. Is there an analogous interpretation for logistic regression?

Well, there is, but it requires a bit of extra work to define some terms so that we can talk sensibly about *what* changes in response to a change in a predictor.

Odds vs Probability

The *odds* of an event E with probability p are given by

$$\text{Odds}(E) = \frac{p}{1-p}.$$

That is, the odds of an event is the ratio of the probability of the event happening to the probability that it doesn't happen. So, for example, if we have an event that occurs with probability $1/2$, the odds of that event are $(1/2)/(1/2) = 1$, which we usually say as "one to one odds". Some of this kind of vocabulary may be familiar to you from sports betting, if you've been to Las Vegas or watched the Kentucky Derby.

Interpreting model coefficients (cont.) I

So, let's suppose that we have a logistic regression model with a single predictor, that takes predictor x and outputs a probability

$$p(x) = \frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}}.$$

and note that

$$1 - p(x) = \frac{\exp\{-(\beta_0 + \beta_1 x)\}}{1 + \exp\{-(\beta_0 + \beta_1 x)\}}$$

The *odds* associated with this probability are

$$\text{Odds}(x) = \frac{p(x)}{1 - p(x)} = \frac{1}{\exp\{-(\beta_0 + \beta_1 x)\}} = \exp\{\beta_0 + \beta_1 x\}.$$

Since the odds can get very large or very small, especially once we start working with small probabilities, it is often much easier to work with the *logarithm* of the odds, usually called the "log-odds" for short. This quantity is especially important in statistics associated with biology

Interpreting model coefficients (cont.) II

(e.g., drug trials and studies of risks associated with diseases). If we look at the *log* odds associated with our logistic regression model,

$$\text{Log-Odds}(x) = \log \text{Odds}(x) = \log \exp\{\beta_0 + \beta_1 x\} = \beta_0 + \beta_1 x.$$

In other words, under our logistic regression model, the "slope" β_1 is the increase in the log-odds of the response being 1 associated with a unit increase in x.

Said yet another way, logistic regression is what happens if we use a linear regression model to predict the log-odds of the event that the response is 1, i.e., the log-odds of the event $Y_i = 1$.

Fitting the model I

We said previously that we were going to ignore, for the time being, the matter of *how* we fit out logistic regression model. Well, let's come back to that question.

Let's start by recalling *ordinary least squares* regression, where we had the following *loss function*:

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2,$$

where $\hat{y}_i = \beta_0 + \beta_1 x_i$ is the predicted y_i based on the coefficients β_0, β_1. This function measures how "wrong" our model is if we use coefficients β_0 and β_1, and we choose these coefficients in such a way as to make thie loss function as small as possible. We used the *square* of the loss because it makes a lot of the math easier.

Now that we're working with logistic regression instead of linear, how do we decide what makes a good choice of coefficient? We *could* try and force a way to use least squares, but things would get a bit complicated.

Fitting the model II

Let's try something different. Our logistic regression model takes a predictor and outputs a probability that the response is 1. That is, for our i-th predictor, our model outputs (we're sticking with the case of one predictor for simplicity, but the idea extends to multiple predictors in the natural way):

$$\Pr[Y_i = 1; X_i = x, \beta_0, \beta_1] = \frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}}.$$

In other words, this is the probability that our model gives to the event that the i-th response is 1. On the other hand, our model assigns probability to the event that the i-th response is 0 given by

$$\begin{aligned}\Pr[Y_i = 0; X_i = x, \beta_0, \beta_1] &= 1 - \Pr[Y_i = 1; X_i = x, \beta_0, \beta_1] \\ &= \frac{\exp\{-(\beta_0 + \beta_1 x)\}}{1 + \exp\{-(\beta_0 + \beta_1 x)\}} \\ &= \frac{1}{1 + \exp\{\beta_0 + \beta_1 x\}}.\end{aligned}$$

Fitting the model III

Now, if the i-th response really is 0, then we want this number to be big. And if the i-th response really is 1, we want $1/(1+\exp\{-(\beta_0+\beta_1 x)\})$ to be big.

Further, we want this pattern to hold for all of our data. That is, we want our model to give a "high probability" to all of our data. If our i-th observation takes the form (x_i, y_i), then the probability of our data under the model, assuming the observations are independent, is

$$\prod_{i=1}^{n} \Pr[Y_i = y_i; X_i = x_i, \beta_0, \beta_1].$$

For one of our data points, using the fact that y_i is either 0 or 1, we can write

$$\Pr[Y_i = y_i; X_i = x_i, \beta_0, \beta_1] = \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}}\right)^{y_i} \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}}\right)^{1-y_i}.$$

So the probability of our whole data set is

$$\prod_{i=1}^{n} \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}}\right)^{y_i} \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}}\right)^{1-y_i}.$$

Fitting the model IV

So this is the probability of all of our data under our model, for a particular choice of parameters β_0 and β_1. We call this the *likelihood* of the data. One way to pick our model coefficients is to choose them so that this quantity is large– if this model is large, that means our data agrees with the model.

So we want to choose β_0 and β_1 to make this probability, the likelihood, large. How do we do that? Well, our likelihood is a product of a bunch of things, and products are hard to work with. Let's take a logarithm. Remember, logs turn products into sums, and sums are easy to work with.

$$\begin{aligned}
&\log \prod_{i=1}^{n} \left(\frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}} \right)^{y_i} \left(\frac{1}{1 + \exp\{\beta_0 + \beta_1 x\}} \right)^{1-y_i} \\
&\quad = \sum_{i=1}^{n} \log \left(\frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}} \right)^{y_i} \left(\frac{1}{1 + \exp\{\beta_0 + \beta_1 x\}} \right)^{1-y_i}
\end{aligned}$$

Fitting the model V

Now, let's notice that an individual term in this sum is a log of a product, and that $\log a^b = b \log a$, so that:

$$\begin{aligned}
&\log \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}} \right)^{y_i} \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}} \right)^{1-y_i} \\
&= \log \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}} \right)^{y_i} + \log \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}} \right)^{1-y_i} \\
&= y_i \log \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}} \right) + (1-y_i) \log \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}} \right)
\end{aligned}$$

So the *logarithm* of our likelihood is

$$\begin{aligned}
&\log \prod_{i=1}^{n} \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}} \right)^{y_i} \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}} \right)^{1-y_i} \\
&= \sum_{i=1}^{n} y_i \log \left(\frac{1}{1+\exp\{-(\beta_0+\beta_1 x)\}} \right) + (1-y_i) \log \left(\frac{1}{1+\exp\{\beta_0+\beta_1 x\}} \right).
\end{aligned}$$

Fitting the model VI

Notice that since the logarithm function is *monotone*, making the likelihood large is the same as making the likelihood large and vice versa.

Logistic regression chooses the coefficients β_0 and β_1 to make this log-likelihood large. Equivalently, we *minimize* the *negative* log likelihood,

$$\ell(\beta_0, \beta_1) = -\sum_{i=1}^{n} \left[y_i \log \left(\frac{1}{1 + \exp\{-(\beta_0 + \beta_1 x)\}} \right) + (1 - y_i) \log \left(\frac{1}{1 + \exp\{\beta_0 + \beta_1 x\}} \right) \right].$$

That is, while our *loss function* for linear regression is the sum of squared errors, our loss function for logistic regression is the *negative log likelihood*.

Unlike in the least squares case, these optimized coefficients do not have nice closed form solutions. Still, when we conduct logistic regression in R, this minimization problem is solved for us using tools from optimization.

A bit of philosophy: maximum likelihood

It turns out that this idea of choosing our parameters to maximize the probability of the data is so popular in statistics that it has a name: *maximum likelihood estimation*. When we need to estimate the value of a parameter, we choose the one that maximizes the likelihood of the data.

Interestingly, in many situations, the *least squares* estimate and the *maximum likelihood* estimate of a parameter are, in fact, the same. In many situations, the sample mean *is* the least squares estimate of our parameter, and it is *also* the estimate that we would obtain if we maximize the likelihood. Examples of this kind of situation include: linear regression, estimating the mean of a normal, and estimating the rate parameter of the Poisson.

You'll see many more connections between these ideas, and establish some of the interesting properties of maximum likelihood estimation in your more advanced statistics classes. For now, let's very quickly establish that the sample mean of the normal is *both* the least squares estimator *and* the maximum likelihood estimator.

Example: mean of a normal

Consider the setting in which we observe data $X_1, X_2, \ldots, X_n$ drawn independently and identically distributed according to a normal distribution with *unknown* mean μ and *known* variance σ^2 (assuming the variance is known is just for the sake of making some things simpler– a similar story is true if we have to estimate the variance, as well).

So let's suppose that we observe $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$. Let's consider two different ways to estimate the mean μ.

Least squares estimation. In least squares, we want to choose the number that minimizes the sum of squares between our estimate and the data:

$$\min_m \sum_{i=1}^{n} (x_i - m)^2.$$

Let's dust off our calculus and solve this. We'll take the derivative and set it equal to zero. First, let's find the derivative:

$$\frac{d}{dm} \sum_{i=1}^{n} (x_i - m)^2 = \sum_{i=1}^{n} \frac{d}{dm} (x_i - m)^2 = 2 \sum_{i=1}^{n} (x_i - m),$$

where we used the fact that the derivative is linear (so the derivative of a sum is the sum of the

Prediction versus feature selection: the two cultures

We have motivated most of our discussion of regression the last few weeks by talking about *prediction*. We see data X_i, and our goal is to predict (i.e., guess) an associated response or label Y_i. A “good” model is one that does well at predicting these labels or responses.

Example: image recognition

Consider once again the problem of detecting whether or not an image contains a cat. We observe (X_i, Y_i) pairs in which each image X_i has an associated label

$$Y_i = \begin{cases} 1 & \text{if image } i \text{ contains a cat} \\ 0 & \text{if image } i \text{ does not contain a cat.} \end{cases}$$

Our goal is then to build a model that takes in an image x and produces a prediction (i.e., a best guess) $\hat{y}$ based on X as to the true label y. A good model is one that correctly guesses y most of the time. Something like

$$\Pr[\hat{y} = y] \text{ is close to } 1$$

Example: housing prices

Consider once again the problem of predicting how much a house will sell for. We get predictors in the form of a vector x containing the house's age, square footage, proximity to parks, quality of the local school system, and so on, and our goal is to predict the price y of that house.

Prediction vs Modeling: what's the goal? I

A "good" model is one whose prediction $\hat{y}$ is "close" to the true value y "on average". i.e., we want to minimize $\mathbb{E}\left(\hat{y} - y\right)^2$

Now, if our goal is *only* to detect whether or not a picture contains a cat, or *only* to predict how much a house will sell for, then that's kind of the end of the story.

But suppose that we have a really good predictor for housing price, and someone asks us "what is it that makes a house expensive?". Just because we have a good *predictive* model doesn't mean that we can answer this question easily.

For example, large complicated neural nets are very good at image detection, but it's very hard to take a trained neural net and figure out what it is about a particular image that causes the neural net to "believe" that there is or isn't a cat in the image. There is no easy way to determine what the "meaningful" or "predictive" features of the image are. Indeed, this problem is a major area of research currently in machine learning and statistics. See here if you're curious to learn more.

Prediction vs Modeling: what's the goal? II

One of the good things about linear and logistic regression is that we can very easily determine which features were "useful" or "meaningful" for prediction– the estimated coefficients and associated p-values tell us quite a lot about which predictors (i.e., "features" in the language of machine learning) are (probably) useful.

Indeed, sometimes the determination of which predictors are "useful" or "meaningful" is the important part of the statistical question.

This tension, between prediction and modeling, was discussed extensively in a famous paper from 2001 by Leo Breiman, titled *Statistical Modeling: The Two Cultures*.

Now, it's not as though there's some huge fight between "prediction people" and "modeling people". We work together all the time! But it's useful to understand the distinction between these two different kinds of problems and to be able to distinguish when one or the other is the more appropriate "hat" to be wearing. Keep an eye out for these kinds of things as you progress in your data science career!

Review

- Relationship between probability, odds and log-odds
- Sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$
- Simple logistic model log-odds$(Y = 1) = \beta_0 + \beta_1 X_1$
- Predicting y from a logistic model
- interpreting the coefficients of a logistic model
- Z tests of significance of logistic coefficients
- fitting a logistic model (maximize log-likelihood / minimize residual deviance)
- likelihood / log likelihood / residual deviance
- confusion matrix for a logistic model, prediction statistics