00:02 Monday 18th April, 2016

Chapter 11

Logistic Regression

11.1 Modeling Conditional Probabilities

So far, we either looked at estimating the conditional expectations of continuous variables (as in regression), or at estimating distributions. There are many situations where however we are interested in input-output relationships, as in regression, but the output variable is discrete rather than continuous. In particular there are many situations where we have binary outcomes (it snows in Pittsburgh on a given day, or it doesn't; this squirrel carries plague, or it doesn't; this loan will be paid back, or it won't; this person will get heart disease in the next five years, or they won't). In addition to the binary outcome, we have some input variables, which may or may not be continuous. How could we model and analyze such data?

We could try to come up with a rule which guesses the binary output from the input variables. This is called **classification**, and is an important topic in statistics and machine learning. However, guessing "yes" or "no" is pretty crude — especially if there is no perfect rule. (Why should there be a perfect rule?) Something which takes noise into account, and doesn't just give a binary answer, will often be useful. In short, we want probabilities — which means we need to fit a stochastic model.

What would be nice, in fact, would be to have conditional distribution of the response Y, given the input variables, $\Pr(Y|X)$. This would tell us about how precise our predictions should be. If our model says that there's a 51% chance of snow and it doesn't snow, that's better than if it had said there was a 99% chance of snow (though even a 99% chance is not a sure thing). We will see, in Chapter 14, general approaches to estimating conditional probabilities non-parametrically, which can use the kernels for discrete variables from Chapter 4. While there are a lot of merits to this approach, it does involve coming up with a model for the joint distribution of outputs Y and inputs X, which can be quite time-consuming.

Let's pick one of the classes and call it "1" and the other "0". (It doesn't matter which is which.) Then Y becomes an **indicator variable**, and you can convince yourself that $\Pr(Y = 1) = \mathbb{E}[Y]$. Similarly, $\Pr(Y = 1|X = x) = \mathbb{E}[Y|X = x]$. (In a phrase, "conditional probability is the conditional expectation of the indicator".)

This helps us because by this point we know all about estimating conditional expectations. The most straightforward thing for us to do at this point would be to pick out our favorite smoother and estimate the regression function for the indicator variable; this will be an estimate of the conditional probability function.

There are two reasons not to just plunge ahead with that idea. One is that probabilities must be between 0 and 1, but our smoothers will not necessarily respect that, even if all the observed y_i they get are either 0 or 1. The other is that we might be better off making more use of the fact that we are trying to estimate probabilities, by more explicitly modeling the probability.

Assume that $\Pr(Y = 1|X = x) = p(x;\theta)$, for some function p parameterized by θ. parameterized function θ, *and* further assume that observations are independent of each other. The the (conditional) likelihood function is

$$\prod_{i=1}^{n} \Pr\left(Y = y_i|X = x_i\right) = \prod_{i=1}^{n} p(x_i;\theta)^{y_i}(1 - p(x_i;\theta))^{1-y_i} \tag{11.1}$$

Recall that in a sequence of Bernoulli trials $y_1,\ldots y_n$, where there is a constant probability of success p, the likelihood is

$$\prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} \tag{11.2}$$

As you learned in basic statistics, this likelihood is maximized when $p = \hat{p} = n^{-1}\sum_{i=1}^{n} y_i$. If each trial had its own success probability p_i, this likelihood becomes

$$\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \tag{11.3}$$

Without some constraints, estimating the "inhomogeneous Bernoulli" model by maximum likelihood doesn't work; we'd get $\hat{p}_i = 1$ when $y_i = 1$, $\hat{p}_i = 0$ when $y_i = 0$, and learn nothing. If on the other hand we assume that the p_i aren't just arbitrary numbers but are linked together, if we *model* the probabilities, those constraints give non-trivial parameter estimates, and let us generalize. In the kind of model we are talking about, the constraint, $p_i = p(x_i;\theta)$, tells us that p_i must be the same whenever x_i is the same, and if p is a continuous function, then similar values of x_i must lead to similar values of p_i. Assuming p is known (up to parameters), the likelihood is a function of θ, and we can estimate θ by maximizing the likelihood. This chapter will be about this approach.

11.2 Logistic Regression

To sum up: we have a binary output variable Y, and we want to model the conditional probability $\Pr(Y = 1|X = x)$ as a function of x; any unknown parameters in the function are to be estimated by maximum likelihood. By now, it will not surprise you to learn that statisticians have approached this problem by asking themselves "how can we use linear regression to solve this?"

1. The most obvious idea is to let $p(x)$ be a linear function of x. Every increment of a component of x would add or subtract so much to the probability. The conceptual problem here is that p must be between 0 and 1, and linear functions are unbounded. Moreover, in many situations we empirically see "diminishing returns" — changing p by the same amount requires a bigger change in x when p is already large (or small) than when p is close to 1/2. Linear models can't do this.

2. The next most obvious idea is to let $\log p(x)$ be a linear function of x, so that changing an input variable *multiplies* the probability by a fixed amount. The problem is that logarithms are unbounded in only one direction, and linear functions are not.

3. Finally, the easiest modification of $\log p$ which has an unbounded range is the **logistic** (or **logit**) **transformation**, $\log \frac{p}{1-p}$. We can make *this* a linear function of x without fear of nonsensical results. (Of course the results could still happen to be *wrong*, but they're not *guaranteed* to be wrong.)

This last alternative is **logistic regression**.

Formally, the logistic regression model is that

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta \tag{11.4}$$

Solving for p, this gives

$$p(x;\beta) = \frac{e^{\beta_0 + x\cdot\beta}}{1+e^{\beta_0+x\cdot\beta}} = \frac{1}{1+e^{-(\beta_0+x\cdot\beta)}} \tag{11.5}$$

Notice that the overall specification is a lot easier to grasp in terms of the transformed probability that in terms of the untransformed probability.[1]

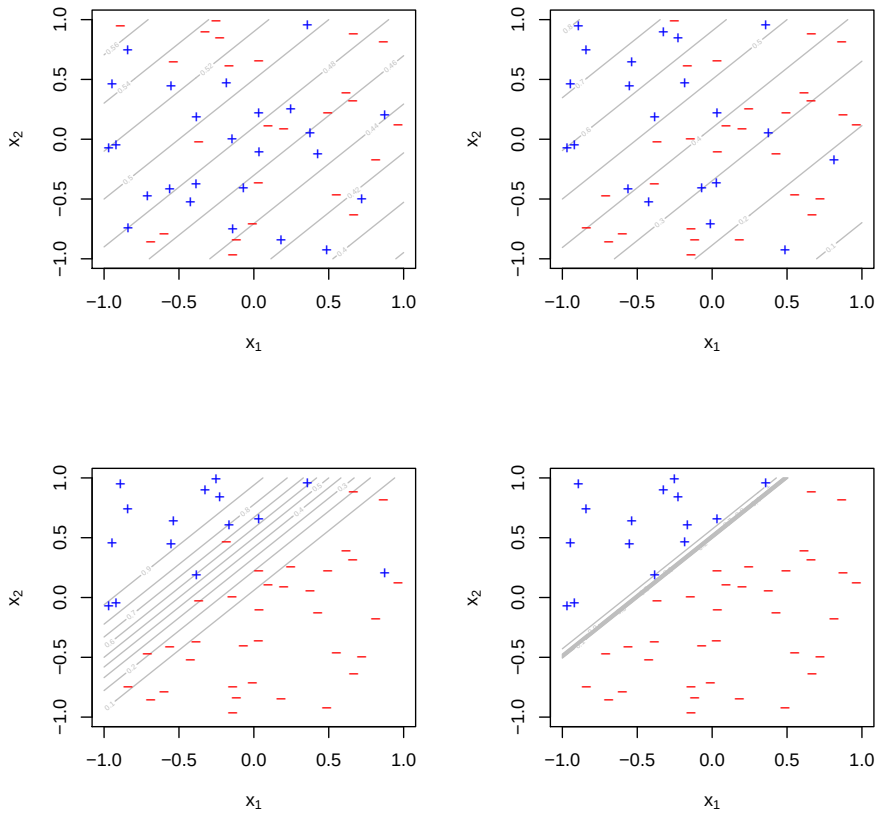
To minimize the mis-classification rate, we should predict $Y = 1$ when $p \geq 0.5$ and $Y = 0$ when $p < 0.5$ (Exercise 1). This means guessing 1 whenever $\beta_0 + x \cdot \beta$ is non-negative, and 0 otherwise. So logistic regression gives us a **linear classifier**. The **decision boundary** separating the two predicted classes is the solution of $\beta_0 + x \cdot \beta = 0$, which is a point if x is one dimensional, a line if it is two dimensional, etc. One can show (exercise!) that the distance from the decision boundary is $\beta_0/||\beta|| + x \cdot \beta/||\beta||$. Logistic regression not only says where the boundary between the classes is, but also says (via Eq. 11.5) that the class probabilities depend on distance from the boundary, in a particular way, and that they go towards the extremes (0 and 1) more rapidly when $||\beta||$ is larger. It's these statements about probabilities which make logistic regression more than just a classifier. It makes stronger, more detailed predictions, and can be fit in a different way; but those strong predictions could be wrong.

Using logistic regression to predict class probabilities is a *modeling choice*, just like it's a modeling choice to predict quantitative variables with linear regression.

[1] Unless you've taken thermodynamics or physical chemistry, in which case you recognize that this is the Boltzmann distribution for a system with two states, which differ in energy by $\beta_0 + x \cdot \beta$.

```
sim.logistic <- function(x, beta.0, beta, bind = FALSE) {
    require(faraway)
    linear.parts <- beta.0 + (x %*% beta)
    y <- rbinom(nrow(x), size = 1, prob = ilogit(linear.parts))
    if (bind) {
        return(cbind(x, y))
    }
    else {
        return(y)
    }
}
plot.logistic.sim <- function(x, beta.0, beta, n.grid = 50, labcex = 0.3, col = "grey",
    ...) {
    grid.seq <- seq(from = -1, to = 1, length.out = n.grid)
    plot.grid <- as.matrix(expand.grid(grid.seq, grid.seq))
    require(faraway)
    p <- matrix(ilogit(beta.0 + (plot.grid %*% beta)), nrow = n.grid)
    contour(x = grid.seq, y = grid.seq, z = p, xlab = expression(x[1]), ylab = expression(x[2]),
        main = "", labcex = labcex, col = col)
    y <- sim.logistic(x, beta.0, beta, bind = FALSE)
    points(x[, 1], x[, 2], pch = ifelse(y == 1, "+", "-"), col = ifelse(y ==
        1, "blue", "red"))
    invisible(y)
}
```

CODE EXAMPLE 25: *Code to simulate binary responses from a logistic regression model, and to plot a 2D logistic regression's probability contours and simulated binary values. (How would you modify this to take the responses from a data frame?*




```
x <- matrix(runif(n = 50 * 2, min = -1, max = 1), ncol = 2)
par(mfrow = c(2, 2))
plot.logistic.sim(x, beta.0 = -0.1, beta = c(-0.2, 0.2))
y.1 <- plot.logistic.sim(x, beta.0 = -0.5, beta = c(-1, 1))
plot.logistic.sim(x, beta.0 = -2.5, beta = c(-5, 5))
plot.logistic.sim(x, beta.0 = -250, beta = c(-500, 500))
```

FIGURE 11.1: *Effects of scaling logistic regression parameters. Values of x_1 and x_2 are the same in all plots ($\sim$ Unif$(-1,1)$ for both coordinates), but labels were generated randomly from logistic regressions with $\beta_0 = -0.1$, $\beta = (-0.2, 0.2)$ (top left); from $\beta_0 = -0.5$, $\beta = (-1,1)$ (top right); from $\beta_0 = -2.5$, $\beta = (-5,5)$ (bottom left); and from $\beta_0 = 2.5 \times 10^2$, $\beta = (-5 \times 10^2, 5 \times 10^2)$. Notice how as the parameters get increased in constant ratio to each other, we approach a deterministic relation between Y and x, with a linear boundary between the classes. (We save one set of the random binary responses for use later, as the imaginatively-named* `y.1`.*)*

In neither case is the appropriateness of the model guaranteed by the gods, nature, mathematical necessity, etc. We begin by positing the model, to get something to work with, and we end (if we know what we're doing) by checking whether it really does match the data, or whether it has systematic flaws.

Logistic regression is one of the most commonly used tools for applied statistics and discrete data analysis. There are basically four reasons for this.

1. Tradition.

2. In addition to the heuristic approach above, the quantity $\log p/(1-p)$ plays an important role in the analysis of contingency tables (the "log odds"). Classification is a bit like having a contingency table with two columns (classes) and infinitely many rows (values of x). With a finite contingency table, we can estimate the log-odds for each row empirically, by just taking counts in the table. With infinitely many rows, we need some sort of interpolation scheme; logistic regression is linear interpolation for the log-odds.

3. It's closely related to "exponential family" distributions, where the probability of some vector v is proportional to $\exp\left\{\beta_0 + \sum_{j=1}^{m} f_j(v)\beta_j\right\}$. If one of the components of v is binary, and the functions f_j are all the identity function, then we get a logistic regression. Exponential families arise in many contexts in statistical theory (and in physics!), so there are lots of problems which can be turned into logistic regression.

4. It often works surprisingly well as a classifier. But, many simple techniques often work surprisingly well as classifiers, and this doesn't really testify to logistic regression getting the probabilities right.

11.2.1 Likelihood Function for Logistic Regression

Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood. For each training data-point, we have a vector of features, x_i, and an observed class, y_i. The probability of that class was either p, if $y_i = 1$, or $1-p$, if $y_i = 0$. The likelihood is then

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i} \tag{11.6}$$

(I could substitute in the actual equation for p, but things will be clearer in a moment if I don't.) The log-likelihood turns products into sums:

$$\begin{aligned}
\ell(\beta_0, \beta) &= \sum_{i=1}^{n} y_i \log p(x_i) + (1-y_i)\log(1-p(x_i)) && (11.7)\\
&= \sum_{i=1}^{n} \log\left(1-p(x_i)\right) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1-p(x_i)} && (11.8)\\
&= \sum_{i=1}^{n} \log\left(1-p(x_i)\right) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) && (11.9)\\
&= \sum_{i=1}^{n} -\log\left(1+e^{\beta_0 + x_i \cdot \beta}\right) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) && (11.10)
\end{aligned}$$

where in the next-to-last step we finally use equation 11.4.

Typically, to find the maximum likelihood estimates we'd differentiate the log likelihood with respect to the parameters, set the derivatives equal to zero, and solve. To start that, take the derivative with respect to one component of β, say β_j.

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_j} &= -\sum_{i=1}^{n} \frac{1}{1+e^{\beta_0 + x_i \cdot \beta}} e^{\beta_0 + x_i \cdot \beta} x_{ij} + \sum_{i=1}^{n} y_i x_{ij} && (11.11)\\
&= \sum_{i=1}^{n} \left(y_i - p(x_i; \beta_0, \beta)\right) x_{ij} && (11.12)
\end{aligned}$$

We are not going to be able to set this to zero and solve exactly. (That's a transcendental equation, and there is no closed-form solution.) We can however approximately solve it numerically.

11.3 Numerical Optimization of the Likelihood

While our likelihood isn't nice enough that we have an explicit expression for the maximum (the way we do in OLS or WLS), it is a pretty well-behaved function, and one which is amenable to lots of the usual numerical methods for optimization. In particular, like most log-likelihood functions, it's suitable for an application of Newton's method. Briefly (see Appendix H.2 for details), Newton's method starts with an initial guess about the optimal parameters, and then calculates the gradient of the log-likelihood with respect to those parameters. It then adds an amount proportional to the gradient to the parameters, moving up the surface of the log-likelihood function. The size of the step in the gradient direction is dictated by the second derivatives — it takes bigger steps when the second derivatives are small (so the gradient is a good guide to what the function looks like), and small steps when the curvature is large.

11.3.1 Iteratively Re-Weighted Least Squares

This discussion of Newton's method is quite general, and therefore abstract. In the particular case of logistic regression, we can make everything look much more like a

good old fashioned statistics problem.

Logistic regression, after all, is a linear model for a transformation of the probability. Let's call this transformation g:

$$g(p) \equiv \log \frac{p}{1-p} \tag{11.13}$$

So the model is

$$g(p) = \beta_0 + x \cdot \beta \tag{11.14}$$

and $Y|X = x \sim \text{Binom}(1, g^{-1}(\beta_0 + x \cdot \beta))$. It seems that what we should want to do is take $g(y)$ and regress it linearly on x. Of course, the variance of Y, according to the model, is going to change depending on x — it will be $(g^{-1}(\beta_0 + x \cdot \beta))(1 - g^{-1}(\beta_0 + x \cdot \beta))$ — so we really ought to do a weighted linear regression, with weights inversely proportional to that variance. Since writing $g^{-1}(\beta_0 + x \cdot \beta)$ is getting annoying, let's abbreviate it by $p(x)$ or just p, and let's abbreviate that variance as $V(p)$.

The problem is that y is either 0 or 1, so $g(y)$ is either $-\infty$ or $+\infty$. We will evade this by using Taylor expansion.

$$g(y) \approx g(p) + (y - p)g'(p) \equiv z \tag{11.15}$$

The right hand side, z will be our *effective* response variable, which we will regress on x. To see why this should give us the right coefficients, substitute for $g(p)$ in the definition of z,

$$z = \beta_0 + x \cdot \beta + (y - p)g'(p) \tag{11.16}$$

and notice that, if we've got the coefficients right, $\mathbb{E}[Y|X = x] = p$, so $(y - p)$ should be mean-zero noise. In other words, when we have the right coefficients, z *is* a linear function of x plus mean-zero noise. (This is our excuse for throwing away the rest of the Taylor expansion, even though we know the discarded terms are infinitely large!) That noise doesn't have constant variance, but we can work it out,

$$\mathbb{V}[Z|X = x] = \mathbb{V}\left[(Y - p)g'(p)|X = x\right] = (g'(p))^2 V(p)\,, \tag{11.17}$$

and so use that variance in weighted least squares to recover β.

Notice that z and the weights both involve the parameters of our logistic regression, through $p(x)$. So having done this once, we should really use the new parameters to update z and the weights, and do it again. Eventually, we come to a fixed point, where the parameter estimates no longer change. This loop — start with a guess about the parameters, use it to calculate the z_i and their weights, regress on the x_i to get new parameters, and repeat — is known as **iterative reweighted least squares** (IRLS or IRWLS), **iterative weighted least squares** (IWLS), etc.

The treatment above is rather heuristic[2], but it turns out to be equivalent to using Newton's method, only with the expected second derivative of the log likelihood, instead of its actual value. This takes a reasonable amount of algebra to show, so

[2]That is, mathematically incorrect.

we'll skip it (but see Exercise 3)[3]. Since, with a large number of observations, the observed second derivative should be close to the expected second derivative, this is only a small approximation.

11.4 Generalized Linear and Additive Models

Logistic regression is part of a broader family of **generalized linear models** (GLMs), where the conditional distribution of the response falls in some parametric family, and the parameters are set by the linear predictor. Ordinary, least-squares regression is the case where response is Gaussian, with mean equal to the linear predictor, and constant variance. Logistic regression is the case where the response is binomial, with n equal to the number of data-points with the given x (usually but not always 1), and p is given by Equation 11.5. Changing the relationship between the parameters and the linear predictor is called changing the **link function**. For computational reasons, the link function is actually the function you apply to the mean response to get back the linear predictor, rather than the other way around — (11.4) rather than (11.5). There are thus other forms of binomial regression besides logistic regression.[4] There is also Poisson regression (appropriate when the data are counts without any upper limit), gamma regression, etc.; we will say more about these in Chapter 12.

In R, any standard GLM can be fit using the (base) `glm` function, whose syntax is very similar to that of `lm`. The major wrinkle is that, of course, you need to specify the family of probability distributions to use, by the `family` option — `family=binomial` defaults to logistic regression. (See `help(glm)` for the gory details on how to do, say, probit regression.) All of these are fit by the same sort of numerical likelihood maximization.

Perfect Classification One caution about using maximum likelihood to fit logistic regression is that it can seem to work badly when the training data *can* be linearly separated. The reason is that, to make the likelihood large, $p(x_i)$ should be large when $y_i = 1$, and p should be small when $y_i = 0$. If β_0, β_0 is a set of parameters which perfectly classifies the training data, then $c\beta_0, c\beta$ is too, for any $c > 1$, but in a logistic regression the second set of parameters will have more extreme probabilities, and so a higher likelihood. For linearly separable data, then, there is no parameter vector which *maximizes* the likelihood, since ℓ can always be increased by making the vector larger but keeping it pointed in the same direction.

You should, of course, be so lucky as to have this problem.

[3]The two key points are as follows. First, the gradient of the log-likelihood turns out to be the sum of the $z_i x_i$. (Cf. Eq. 11.12.) Second, take a single Bernoulli observation with success probability p. The log-likelihood is $Y \log p + (1-Y)\log 1 - p$. The first derivative with respect to p is $Y/p - (1-Y)/(1-p)$, and the second derivative is $-Y/p^2 - (1-Y)/(1-p)^2$. Taking expectations of the second derivative gives $-1/p - 1/(1-p) = -1/p(1-p)$. In other words, $V(p) = -1/\mathbb{E}\left[\ell''\right]$. Using weights inversely proportional to the variance thus turns out to be equivalent to dividing by the expected second derivative. But gradient divided by second derivative is the increment we use in Newton's method, QED.

[4]My experience is that these tend to give similar error rates as classifiers, but have rather different guesses about the underlying probabilities.