Win-Vector Blog

The Win-Vector LLC data science blog

The Simpler Derivation of Logistic Regression

September 14, 2011 Nina Zumel Expository Writing, Pragmatic Machine Learning, Statistics, Statistics To English Translation, Tutorials likelihood, log-likelihood, Logistic Regression, newton's method, Statistics

Logistic regression is one of the most popular ways to fit models for categorical data, especially for binary response data. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear regression, logistic regression can directly predict probabilities (values that are restricted to the (0,1) interval); furthermore, those probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes. Logistic regression preserves the marginal probabilities of the training data. The coefficients of the model also provide some hint of the relative importance of each input variable.

While you don't have to know how to derive logistic regression or how to implement it in order to use it, the details of its derivation give important insights into interpreting and troubleshooting the resulting models. Unfortunately, most derivations (like the ones in [Agresti, 1990] or [Hastie, et.al, 2009]) are too terse for easy comprehension. Here, we give a derivation that is less terse (and less general than Agresti's), and we'll take the time to point out some details and useful facts that sometimes get lost in the discussion.

To make the discussion easier, we will focus on the binary response case. We assume that the case of interest (or "true") is coded to 1, and the alternative case (or "false") is coded to 0.

The logistic regression model assumes that the log-odds of an observation y can be expressed as a linear function of the K input variables $\mathbf{x}$:

$$\log \frac{P(\mathbf{x})}{1-P(\mathbf{x})} = \sum_{j=0}^{K} b_j x_j$$

Here, we add the constant term b_0, by setting $x_0 = 1$. This gives us K+1 parameters. The left hand side of the above equation is called the *logit* of P (hence, the name logistic regression).

Let's take the exponent of both sides of the logit equation.

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = \exp(\sum_{j=0}^{K} b_j x_j)$$

$$= \prod_{j=0}^{K} \exp(b_j x_j)$$

This immediately tells us that logistic models are multiplicative in their inputs (rather than additive, like a linear model), and it gives us a way to interpret the coefficients. The value $\exp(b_j)$ tells us how the odds of the response being "true" increase (or decrease) as x_j increases by one unit, all other things being equal. For example, suppose $b_j = 0.693$. Then $\exp(b_j) = 2$. If x_j is a numerical variable (say, age in years), then every year's increase in age doubles the odds of the response being true — all other things being equal. If x_j is a binary variable (say, sex, with female coded as 1 and male as 0), then if the subject is female, then the response is two times more likely to be true than if the subject is male, all other things being equal.

We can also invert the logit equation to get a new expression for P(x):

$$P(\mathbf{x}) = \frac{\exp z}{1+\exp z},$$

$$z = \sum_{j=0}^{K} b_j x_j$$

The right hand side of the top equation is the sigmoid of z, which maps the real line to the interval (0, 1), and is approximately linear near the origin. A useful fact about P(z) is that the derivative P'(z) = P(z) (1 – P(z)). Here's the derivation:

$$P(z) = \frac{\exp z}{1+\exp z} = (\exp z)(1+\exp z)^{-1}$$

$$P'(z) = (\exp z)(1+\exp z)^{-1} + (\exp z)(-1)(1+\exp z)^{-2}(\exp z)$$

(by chain rule)

$$= \frac{(\exp z)(1+\exp z)}{(1+\exp z)^2} - \frac{(\exp z)^2}{(1+\exp z)^2}$$

$$= \frac{\exp z}{(1+\exp z)^2}$$

$$= \frac{\exp z}{1+\exp z} \cdot \frac{1}{1+\exp z}$$

$$= P(z)(1-P(z))$$

Later, we will want to take the gradient of P with respect to the set of coefficients **b**, rather than *z*. In that case, P'(*z*) = P(*z*) (1 – P(*z*))*z*‘, where ‘ is the gradient taken with respect to **b**.

The solution to a Logistic Regression problem is the set of parameters **b** that maximizes the likelihood of the data, which is expressed as the product of the predicted probabilities of the N individual observations.

$$\mathrm{L}(X|P) = \prod_{i=1,y_i=1}^{N} P(\mathbf{x}_i) \prod_{i=1,y_i=0}^{N} (1-P(\mathbf{x}_i))$$

(*X*, *y*) is the set of observations; *X* is a K+1 by N matrix of inputs, where each column corresponds to an observation, and the first row is **1**; *y* is an N-dimensional vector of responses; and ($\mathbf{x}_i$, y_i) are the individual observations.

It’s generally easier to work with the log of this expression, known (of course) as the log-likelihood.

$$\mathcal{L}(X|P) = \sum_{i=1,y_i=1}^{N} \log P(\mathbf{x}_i) + \sum_{i=0,y_i=0}^{N} \log(1-P(\mathbf{x}_i))$$

Maximizing the log-likelihood will maximize the likelihood. As a side note, the quantity −2*log-likelihood is called the *deviance* of the model. It is analogous to the residual sum of squares (RSS) of a linear model. Ordinary least squares minimizes RSS; logistic regression minimizes deviance. A useful goodness-of-fit heuristic for a logistic regression model is to compare the deviance of the model with the so-called *null deviance*: the deviance of the constant model that returns only the global response probability for every data point. One minus the ratio of deviance to null deviance is sometimes called *pseudo-R^2*, and is used the way one would use R^2 to evaluate a linear model.

$$\text{pseudo-}R^2 = 1 - \frac{\text{deviance}}{\text{null deviance}}$$

Traditional derivations of Logistic Regression tend to start by substituting the logit function directly into the log-likelihood equations, and expanding from there. The derivation is much simpler if we don't plug the logit function in immediately. To maximize the log-likelihood, we take its gradient with respect to **b**:

$$\nabla_{\mathbf{b}}\mathcal{L} = \sum_{\substack{i=0 \\ y_i=1}}^{N} \frac{P_i'}{P_i}\mathbf{x}_i - \sum_{\substack{i=0 \\ y_i=0}}^{N} \frac{P_i'}{1-P_i}\mathbf{x}_i$$

where P_i is shorthand for $P(x_i)$. The maximum occurs where the gradient is zero.

We can expand this equation further, when we remember that P' = P(1-P):

$$\begin{aligned}\nabla_{\mathbf{b}}\mathcal{L} &= \sum_{\substack{i=i \\ y_i=1}}^{N} \frac{P_i(1-P_i)}{P_i}\mathbf{x}_i - \sum_{\substack{i=1 \\ y_i=0}}^{N} \frac{P_i(1-P_i)}{1-P_i}\mathbf{x}_i \\ &= \sum_{\substack{i=1 \\ y_i=1}}^{N} (1-P_i)\mathbf{x}_i - \sum_{\substack{i=1 \\ y_i=0}}^{N} P_i\mathbf{x}_i \\ &= \sum_{i=1}^{N} \left[y_i(1-P_i) - (1-y_i)P_i\right]\mathbf{x}_i\end{aligned}$$

The last line merges the two cases (y_i = 1 and y_i = 0) into a single sum. We can now cancel terms and set the gradient to zero. This gives us the set of simultaneous equations that are true at the optimum:

$$\sum_{i=1}^{N} y_i\mathbf{x}_i - P_i\mathbf{x}_i = \mathbf{0}$$

Notice that the equations to be solved are in terms of the probabilities P (which are a function of **b**), not directly in terms of the coefficients **b** themselves. This means that logistic models are coordinate-free: for a given set of input variables, the probabilities returned by the model will be the same even if the variables are shifted, combined, or rescaled. Only the values of the coefficients will change.

The other thing to notice from the above equations is that the sum of probability mass across each coordinate of the $\mathbf{x}_i$ vectors is equal to the count of observations with that coordinate value for which the response was true. For example, suppose the jth input variable is 1 if the subject is female, 0 if the subject is male. Then

$$\sum_{i=1}^{N} y_i \mathbf{x}_{ij} - P_i \mathbf{x}_{ij} = 0$$

$$\sum_{\substack{i=1 \\ x_{ij}=1}}^{N} y_i - P_i = 0$$

$$\sum_{\substack{i=1 \\ x_{ij}=1}}^{N} y_i = \sum_{\substack{i=1 \\ x_{ij}=1}}^{N} P_i$$

In other words, the summed probability mass for the female subjects equals the count of female subjects with the response "true". It is also true that the sum of all the probability mass over the entire training set will equal the number of "true" responses in the training set. This is what we mean when we say that logistic regression preserves the marginal probabilities of the training data.

Solving for the Coefficients

The most straightforward way to solve for the coefficients **b** is Newton's method. The Fisher scoring method that is used in most off-the-shelf implementations is a more general variation of Newton's method; it works on the same principles. We will describe solving for the coefficients using Newton's method.

Suppose you have a vector valued function **f**: **y** = **f(b)**. You want to find the value $\mathbf{b}_{opt}$ such that $\mathbf{f(b)}_{opt} = \mathbf{0}$. Assuming that we start with an initial guess $\mathbf{b}_0$, we can take the Taylor expansion of **f** around $\mathbf{b}_0$:

$$\mathbf{f}(\mathbf{b_0} + \mathbf{\Delta}) \approx \mathbf{f}(\mathbf{b}_0) + \mathbf{f}'(\mathbf{b}_0)\mathbf{\Delta}$$

Here, **f**' is a matrix; it is the Jacobean of first derivatives of **f** with respect to **b**. Setting the left hand side to zero, we can solve for Δ as

$$\Delta_0 = -[\mathbf{f}'(\mathbf{b}_0)]^{-1}\mathbf{f}(\mathbf{b}_0)$$

We then update our estimate for **b**:

$$\mathbf{b}_1 = \mathbf{b}_0 + \mathbf{\Delta}_0$$

and iterate until convergence.

In our case, **f** is the gradient of the log-likelihood, and its Jacobean is the Hessian (the matrix of second derivatives) of the log-likelihood function.

$$\begin{aligned} H &= \frac{\partial}{\partial \mathbf{b}} \nabla_{\mathbf{b}} \mathcal{L} \\ &= -\sum_{i=1}^{N} \mathbf{x}_i \nabla_{\mathbf{b}} P_i \\ &= -\sum_{i=1}^{N} \mathbf{x}_i P_i (1 - P_i) \mathbf{x}_i^T \\ &\equiv \mathbf{X} \mathbf{W} \mathbf{X}^T \end{aligned}$$

where **W** is a diagonal matrix of the derivatives P'_i, and the ith column of **X** corresponds to the ith observation. So we can solve for Δ at each iteration as

$$\boldsymbol{\Delta}_k = (\mathbf{X} \mathbf{W}_k \mathbf{X}^T)^{-1} \mathbf{X} (\mathbf{y} - \mathbf{P}_k)$$

where **W** is the current matrix of derivatives, **y** is the vector of observed responses, and $\mathbf{P}_k$ is the vector of probabilities as calculated by the current estimate of **b**.

Compare this to the solution of a linear regression:

$$\begin{aligned} \mathbf{y} &= \mathbf{X}^T \mathbf{b} \\ \mathbf{X}\mathbf{y} &= \mathbf{X}\mathbf{X}^T \mathbf{b} \\ \mathbf{b} &= (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X}\mathbf{y} \end{aligned}$$

Comparing the two, we can see that at each iteration, Δ is the solution of a weighted least square problem, where the "response" is the difference between the observed response and its current estimated probability of being true. This is why the technique for solving logistic regression problems is sometimes referred to as *iteratively re-weighted least squares*. Generally, the method does not take long to converge (about 6 or so iterations).

Thinking of logistic regression as a weighted least squares problem immediately tells you a few things that can go wrong, and how. For example, if some of the input variables are correlated, then the Hessian **H** will be ill-conditioned, or even singular. This will result in large error bars (or "loss of significance") around the estimates of certain coefficients. It can also result in coefficients with excessively large magnitudes, and often the wrong sign. If an input perfectly predicts the response for some subset of the data (at no penalty on the rest of the data), then the term $P_i (1 - P_i)$ will be driven to zero for that subset, which will drive the coefficient for that input to infinity (if the input perfectly predicted all the data, then the residual $(\mathbf{y} - \mathbf{P}_k)$ has already gone to zero, which means that you are already at the optimum).

On the other hand, the least squares analogy also gives us the solution to these problems: *regularized regression*, such as lasso or ridge. Regularized regression

penalizes excessively large coefficients, and keeps them bounded. If you are implementing your own logistic regression procedure, rather than using a package, then it is straightforward to implement a regularized least squares for the iteration step (as Win-Vector has done). But even if you are using an off-the-shelf implementation, the above discussion will help give you a sense of how to interpret the coefficients of your model, and how to recognize and troubleshoot some issues that might arise.

Conclusion

Here is what you should now know from going through the derivation of logistic regression step by step:

– Logistic regression models are multiplicative in their inputs.

– The exponent of each coefficient tells you how a unit change in that input variable affects the odds ratio of the response being true.

– Logistic regression is coordinate-free: translations, rotations, and rescaling of the input variables will not affect the resulting probabilities.

– Logistic regression preserves the marginal probabilities of the training data.

– Overly large coefficient magnitudes, overly large error bars on the coefficient estimates, and the wrong sign on a coefficient could be indications of correlated inputs.

– Coefficients that tend to infinity could be a sign that an input is perfectly correlated with a subset of your responses. Or put another way, it could be a sign that this input is only really useful on a subset of your data, so perhaps it is time to segment the data.

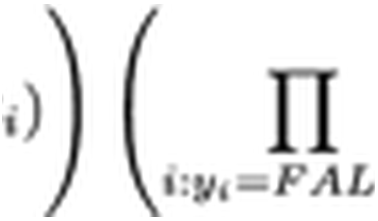
– Pseudo-R^2 is a useful goodness-of-fit heuristic.

References

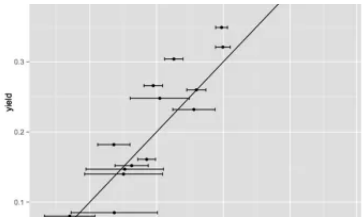
[Agresti, 1990] Agresti, A. (1990). Categorical Data Analysis.

[Hastie, et.al, 2009] Hastie, T., R. Tibshirani, and J. Friedman (2009). The Elements of Statistical Learning, 2nd Edition.

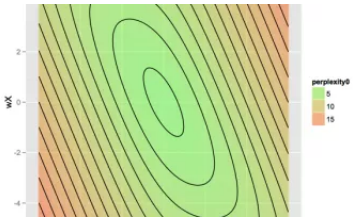
RELATED



What does a generalized linear model do?
In "data science"



Generalized linear models for predicting rates
In "data science"



How robust is logistic regression?
In "data science"

September 14, 2011 Nina Zumel Expository Writing, Pragmatic Machine Learning, Statistics, Statistics To English Translation, Tutorials likelihood, log-likelihood, Logistic Regression, newton's method, Statistics

4 thoughts on “The Simpler Derivation of Logistic Regression”

Rama

September 16, 2011 at 6:42 am

Nice post!

Clearest derivation of LR that I have come across. Neat how the coordinate-freeness and marginal-probability-preservation properties of LR elegantly “fell out” of the derivation.

Rama

September 16, 2011 at 7:53 am

Oh, one other thing.

> Coefficients that tend to infinity could be a sign that an input is perfectly correlated with a subset of your responses. Or put another way, it could be a sign that this input is only really useful on a subset of your data, so perhaps it is time to segment the data. <

Running a (short) decision tree on the data can efficiently uncover such inputs.

Nina Zumel

September 16, 2011 at 9:07 am

@Rama Great suggestion about the decision tree. Thanks for your comments.

I've used decision trees/stumps as pre-processing for regression in a few different ways — someday I'll have to put them all together in article. :)

Barak A. Pearlmutter

October 6, 2011 at 6:14 am

Nice post.

Note that P(z) = exp z / (1 + exp z)
= (exp z / (1 + exp z))(exp -z/exp -z)
= 1 / (1 + exp -z)

This form is more common in the MLP literature, and is a little easier to deal with sometimes because z appears only once. E.g., it is a little easier to solve for z given P.

Comments are closed.