The equivalence of logistic regression and maximum entropy models

John Mount[*]

September 23, 2011

Abstract

As our colleague so aptly demonstrated (http://www.win-vector.com/blog/2011/09/the-simpler-derivation-of-logistic-regression/ (link)) there is one derivation of Logistic Regression that is particularly beautiful. It is not as general as that found in Agresti[Agresti, 1990] (which deals with generalized linear models in their full generality), but gets to the important balance equations very quickly. We will pursue this further to re-derive multi-category logistic regression in both its standard (sigmoid) phrasing and also in its equivalent maximum entropy clothing.

It is well known that logistic regression and maximum entropy modeling are equivalent (for example see [Klein and Manning, 2003])- but we will show that the simpler derivation already given is a very good way to demonstrate the equivalence (and points out that logistic regression is actually special- not just one of many equivalent GLMs).

1 Overview

We will proceed as follows:

1. This outline.
2. Introduce a simplified machine learning problem and some notation.
3. Re-discover logistic regression by a simplified version of the standard derivations.
4. Re-invent logistic regression by using the maximum entropy method.
5. Draw some conclusions.

2 Notation

Suppose our machine learning input is a sequence of real vectors of dimension n (where we have pre-processed in the machine learning tricks of converting categorical variables into indicators over levels and adding a constant variable to our representation).

Our notation will be as follows:

1. $x(1) \cdots x(m)$ will denote our input data. Each $x(i)$ is a vector in $\mathbb{R}^n$. We will use the function of i notation to denote which specific example we are working with. We will also use the variable j to denote which of the n coordinates or parameters we are interested in (as in $x(i)_j$).

[*]email: mailto:jmount@win-vector.com web: http://www.win-vector.com/

2. $y(1) \cdots y(m)$ will denote our known training outcomes. The $y(i)$ take values from the set $\{1 \cdots k\}$. For standard two class logistic regression we have $k = 2$ (the target class and its complement). There is no implied order relation between the classes- each class is just using a different symbol. We will use the variables u and v to indicate which class/category/outcome we are working with.

3. $\pi()$ will be the modeled (or learned) probability function we are solving for. $\pi() : \mathbb{R}^n \to \mathbb{R}^k$ or in English: $\pi()$ is a function mapping from the space the $x(i)$ are in into vectors indexed by the categories. In fact $\pi(x)_u$ will denote the probability $\pi()$ assigns in input situation x to category u.

4. "$\mathrm{A}(u, v)$" will be a notational convenience called "an indicator function." It is defined as being the unique function that has $\mathrm{A}(u, v) = 1$ if $u = v$ and 0 otherwise. We will use $\mathrm{A}(,)$ to check if a given $y(i)$ (a known training outcome) is assigned to the category u that we are at the time working on (so we will write expressions like $\mathrm{A}(u, y(i))$).

Our notation is designed to be explicit (at the cost of some wordiness). For example we will do most work in scalar notation (using partial derivatives) instead of the much more concise gradient operator notation.

Our goal from this training data is to learn a function such that $f(x(i)) \approx y(i)$ (for all i). Actually we will do a bit more we will develop an estimate function $\pi() : \mathbb{R}^n \to \mathbb{R}^k$ such that: $\pi(x(i))_v$ is a modeled estimate of the probability that $y(i) = v$.

$\pi()$ should have the following properties:

1. $\pi(x)_v \geq 0$ always
2. $\sum_{v=1}^{k} \pi(x)_v = 1$ always
3. $\pi(x(i))_{y(i)}$ tends to be large.

The first two requirements state that $\pi()$ looks like a probability distribution over the possible outcomes and the last states that $\pi()$ is a good prediction (matches our training data). In the next two sections we will formalize what "good" is.

Everything we are going to do (except appealing to the Karush-Kuhn-Tucker theorem) is elementary. We use the slightly cumbersome partial derivative notation (in favor of the more terse gradient vector notation) to preserve the elementary nature of the derivations.

3 Standard Logistic Regression

In standard logistic regression we usually have $k = 2$ (one class is the class we are trying to classify into and the other class is taken as its complement). And in this framework $\pi()$ is chosen out of the blue to be of the form:

$$\pi(x)_1 = \frac{e^{\lambda \cdot x}}{e^{\lambda \cdot x} + 1} \quad (1)$$

$$\pi(x)_2 = 1 - \pi(x)_1 \quad (2)$$

where λ is a vector in $\mathbb{R}^n$. You can check that this form guarantees that our $\pi()$ looks like a probability distribution. Also our model is completely specified by λ, so it is just a matter of picking λ that gives a best estimate. We also note: in this form $\pi(x)_1$ is the so-called "sigmoid function" which is the inverse of the logit-function (hence the name logistic regression). This also hints that logistic regression is equivalent to a single layer neural net with sigmoid non-linearity (though logistic regression fitting by Fisher scoring is a bit more sophisticated than neural net back-prop training).

We are actually going to suggest a different way to write $\pi()$ that has more notational symmetry (and generalizes beyond $k = 2$ for free). Define:

$$\pi(x)_v = \frac{e^{\lambda_v \cdot x}}{\sum_{u=1}^{k} e^{\lambda_u \cdot x}}. \tag{3}$$

In this case λ is now a k by n matrix with one set of values per category we are trying to predict. Our original form is gotten by forcing λ_2 to be the all zero vector. $\pi()$ of this form has two properties that are very useful in deriving logistic regression:

$$\begin{aligned} \frac{\partial}{\partial \lambda_{v,j}} \pi(x)_v &= x_j \pi(x)_v (1 - \pi(x)_v) && (4) \\ \frac{\partial}{\partial \lambda_{u,j}} \pi(x)_v &= -x_j \pi(x)_v \pi(x)_u \text{ (when } u \neq v) && (5) \end{aligned}$$

Let us go back to our requirement that $\pi(x(i))_{y(i)}$ be large. We can formalize that as maximizing the product:

$$\prod_{i=1}^{m} \pi(x(i))_{y(i)} \tag{6}$$

which means we are maximizing the simultaneous "likelihood" the model assigns to the training data (treating data as independent conditioned on the model parameters). This makes sense- the model should not consider the data it was trained on as unlikely.

Equivalently we could maximize a different function (but one that has its maximum in the same place) the log-likelihood:

$$f(\lambda) = \sum_{i=1}^{m} \log(\pi(x(i))_{y(i)}) \tag{7}$$

(remember $\pi(x)$ is a function of x and λ).

We want a λ such that $f()$ is maximized. Now in this notation we have no constraints on λ so we know at any maximal point we will have the derivatives of $f()$ with respect to $\lambda_{u,j}$ equal to zero for all u,j. It will further turn out that under fairly mild assumptions about the training data that the point that all the partial derivatives are zero is unique- so therefore it is the unique global maximum. So we solve for the λ where all the partial derivatives are zero.

$$\begin{aligned}
\frac{\partial}{\partial \lambda_{u,j}} f(\lambda) &= \frac{\partial}{\partial \lambda_{u,j}} \sum_{i=1}^{m} \log(\pi(x(i))_{y(i)}) && (8)\\
&= \sum_{i=1}^{m} \frac{1}{\pi(x(i))_{y(i)}} \frac{\partial}{\partial \lambda_{u,j}} \pi(x(i))_{y(i)} && (9)\\
&= \sum_{\substack{i=1,\\ y(i)=u}}^{m} \frac{1}{\pi(x(i))_{u}} \frac{\partial}{\partial \lambda_{u,j}} \pi(x(i))_{u} && (10)\\
&\quad + \sum_{\substack{i=1,\\ y(i)\neq u}}^{m} \frac{1}{\pi(x(i))_{y(i)}} \frac{\partial}{\partial \lambda_{u,j}} \pi(x(i))_{y(i)} \\
&= \sum_{\substack{i=1,\\ y(i)=u}}^{m} \frac{1}{\pi(x(i))_{u}} x(i)_j \pi(x(i))_u (1 - \pi(x(i))_u) && (11)\\
&\quad - \sum_{\substack{i=1,\\ y(i)\neq u}}^{m} \frac{1}{\pi(x(i))_{y(i)}} x(i)_j \pi(x(i))_{y(i)} \pi(x(i))_u \\
&= \sum_{\substack{i=1,\\ y(i)=u}}^{m} x(i)_j (1 - \pi(x(i))_u) && (12)\\
&\quad - \sum_{\substack{i=1,\\ y(i)\neq u}}^{m} x(i)_j \pi(x(i))_u \\
&= \sum_{\substack{i=1,\\ y(i)=u}}^{m} x(i)_j - \sum_{i=1}^{m} x(i)_j \pi(x(i))_u && (13)
\end{aligned}$$

And since this quantity is supposed to equal zero we learn the relation:

$$\sum_{i=1}^{m} \pi(x(i))_u x(i)_j = \sum_{\substack{i=1,\\ y(i)=u}}^{m} x(i)_j \text{ (for all } u, j). \tag{14}$$

These equations are remarkable. They say: the sum of any coordinate (j) of the $x(i)$'s from all the training data in a particular category u is equal the sum of probability mass the model places in that coordinate summed across all data (for all u, j). Summaries of the training data are preserved by the model.

Using $\mathrm{A}(u, y(i))$ we can re-write equation 14 as:

$$\sum_{i=1}^{m} \pi(x(i))_u x(i)_j = \sum_{i=1}^{m} \mathrm{A}(u, y(i)) x(i)_j \text{ (for all } u, j). \tag{15}$$

This means the best chosen value of λ creates a probability function $\pi(x(i))_u$ that behaves very much like the indicator function $\mathrm{A}(u, y(i))$ (simultaneously for all variables and categories when summed over i).

We will call equations 14 and 15 the "balance equations" (drawing an analogy to the "detail balance equations" from Markov chain theory).

Also notice that the model parameters λ do not appear explicitly in these equations (they are implicit $\pi()$ but are not seen explicitly in the balance equations like the $x(i)$ do). This means we are in some sense "coordinate free" the result we get depends only on the variables we have decided to measure/model and not on how we parameterized $\pi()$ in terms of λ.

Solving this system of $k \times n$ equations involving our $k \times n$ unknown λ's is slightly tricky as $\pi()$ depends on the λ's in a non-linear fashion. However, we can take derivatives of these sets of equations (compute Jacobian of our system of equations, which is essentially the Hessian of $f()$) easily. Then we can apply one of Newton's Method, Fisher Scoring or Iteratively re-Weighted Least Squares to find a λ that satisfies all of our equations very quickly.[1]

From a logistic regression point of view we are done. The lucky form of the sigmoid function (as pulled out of thin air in equations 1 and 3) had such a simple derivative structure that we could derive logistic regression just by re-arranging a few sums.[2] But what if you didn't think of the sigmoid?

4 Maximum Entropy

In the equivalent maximum entropy derivation of logistic regression you don't have to cleverly guess the sigmoid form. Instead you assume you want a balance equation like equation 15 to be true and you can, without needing any luck, solve for the necessary form for $\pi()$.

Start as follows. Assume nothing about $\pi()$- it could be any sort of hideously complicated function. Just start listing things you would like to be true about it (and don't worry if you are asking for too many things to be true or too few things to be true). Suppose we require $\pi()$ to obey the following relations:

$$\begin{aligned} \pi(x)_v &\geq 0 \text{ always} && (16)\\ \sum_{v=1}^{k} \pi(x)_v &= 1 \text{ always} && (17)\\ \sum_{i=1}^{m} \pi(x(i))_u x(i)_j &= \sum_{i=1}^{m} \mathrm{A}(u, y(i)) x(i)_j \text{ (for all } u, j) && (18) \end{aligned}$$

The first two conditions are needed for $\pi()$ to behave like a probability and the third we can think of as saying $\pi(x(i))_u$ should well approximate the category indicator $\mathrm{A}(u, y(i))$ on our training data. Now we have not yet limited the complexity of $\pi()$ which is something we typically do to make sure $\pi()$ is not some horrendous function that is impossible to implement and over-fits the training data in a non-useful fashion. One particularly bad $\pi()$ (which we do not want) is anything like:

$$\begin{aligned} \mathrm{match}(x) &= \text{minimal } i \text{ from } 1 \cdots n \text{ such that } x(i) = x \text{ or } 0 \text{ if there is no such } i.\\ \mathrm{pick}(i) &= y(i) \text{ if } 1 \leq i \leq n \text{ and } 1 \text{ otherwise.}\\ \pi(x) \text{ such that } & \pi(x)_{\mathrm{pick}(\mathrm{match}(x))} = 1 \text{ and } 0 \text{ in all other coordinates.} \end{aligned}$$

This is an evil useless cousin of a the (useful) nearest neighbor model. It may get exact items from training right- but moves to zero once you introduce a novel point. What we want is some notion of continuity, smoothness, minimum description length, Occam's razor or low complexity. In information theory the standard thing to try is maximizing the entropy (which in turn minimizes the over-fitting).[Cover and Thomas, 1991] The entropy of $\pi()$ would be defined as:

$$-\sum_{v=1}^{k} \sum_{i=1}^{m} \pi(x(i))_v \log(\pi(x(i))_v) \qquad (19)$$

[1]Newton's zero finding method is the easiest to derive, it is a remarkable property of GLMs that this method is in this case almost indistinguishable from Fisher Scoring or Iteratively re-Weighted Least Squares.

[2]You can also derive the same update rules for more general GLMs, but the derivation requires slightly more cleverness.

This equation isn't pulled out of thin air as it is the central equation of information theory. It is maximized for simple $\pi()$ (like the constant function $\pi(x) = 1/k$ independent of x).

So we write our problem as a constrained optimization problem: find a function $\pi()$ maximizing the equation 19 obeying equations 16,17,18.

Despite being the simplest form equation 16 is the hardest to work with (because it is an inequality instead of an equality) so leave it out for now and hope that what we find can be shown to satisfy equation 16 (which is how things turn out). Also notice we are not assuming any form of $\pi()$, so we are searching the space of all functions (not just a parameterized sub-space). We will actually derive the form instead of having to guess or know it.

In constrained optimization the standard technique is to set up the Lagrangian. This is just a sum over all constraints and the objective function[3]. Our Lagrangian can be taken to be:

$$\begin{aligned} L &= \sum_{j=1}^{n}\sum_{v=1}^{k}\lambda_{v,j}\left(\sum_{i=1}^{m}\pi(x(i))_v x(i)_j - \mathrm{A}(v,y(i))x(i)_j\right) \\ &\quad + \sum_{v=1}^{k}\sum_{i=1}^{m}\beta_i\left(\pi(x(i))_v - 1\right) \\ &\quad - \sum_{v=1}^{k}\sum_{i=1}^{m}\pi(x(i))_v\log(\pi(x(i))_v) \end{aligned} \tag{20}$$

What we have done (and this is a standard and re-usable technique) is formed L as a sum. For each equation we wanted to enforce we added a corresponding term into our sum multiplied by a new free (note-keeping) variable (the λ's and β's). The objective function is added in without a variable[4]. The signs and variable names are all arbitrary- but we have picked λ to be suggestive (though the technique would work the same if we picked different signs and names).

The subtle point is (and it is a consequence of the Karush-Kuhn-Tucker theorem) is that certain things that are true about the optimal solution are also true about the Lagrangian. In particular the partial derivatives we expect to vanish for an unconstrained optimization problem do in fact vanish for the Lagrangian at the optimum. So the unknown optimal $\pi()$ is such that:

$$\frac{\partial}{\partial\pi(x(i))_u}L = 0 \text{ (for all } i,u). \tag{21}$$

Note that we are treating $\pi()$ almost as lookup table in that we are assuming we can set $\pi(x(i))$ almost independently for each i and further assuming the formal derivatives don't interact for different i. But we also are strict in that we are enforcing a derivative condition at each and every data point. If we have a lot of data (i.e. m is large and the points are diverse) we get a lot of conditions. This severely limits the unknown structure of the optimal $\pi()$ despite the apparent freedom of the formal analysis.

Let us look at $\frac{\partial}{\partial\pi(x(i))_u}L$:

$$\frac{\partial}{\partial\pi(x(i))_u}L = \lambda_u \cdot x(i) + \beta_i - \log(\pi(x(i))_u) - 1 \tag{22}$$

Anything that does not directly have a $\pi(x(i))_u$ goes to zero and we applied the standard rules of derivatives. As we said we expect these derivatives to all be zero for an optimal $\pi()$ we get a huge

[3]Inequality constraints can also be in the Lagrangian, but this requires introducing more variables.

[4]All terms should have a free variable- but by scaling we can leave one variable off- so we left it off the objective function since it is unique.

family of equations:

$$\lambda_u \cdot x(i) + \beta_i - \log(\pi(x(i))_u) - 1 = 0 \tag{23}$$

for all i, u. Or re-arranging a bit:

$$\pi(x(i))_u = e^{\lambda_u \cdot x(i) + \beta_i - 1}. \tag{24}$$

This is starting to look promising. For example we now can see $\pi() \geq 0$ everywhere (a consequence of the exponential form). We can now start to solve for our free variables β and λ which now parameterize the family of forms $\pi()$ we need to look at to find an optimal solution. By equation 17 we know we want to enforce:

$$\sum_{v=1}^{k} e^{\lambda_v \cdot x(i) + \beta_i - 1} = 1 \tag{25}$$

So

$$e^{\beta} = 1 / \sum_{v=1}^{k} e^{\lambda_v \cdot x(i) - 1} \tag{26}$$

And we can plug this β back into $\pi()$ and simplify to get:

$$\pi(x)_u = \frac{e^{\lambda_u \cdot x}}{\sum_{v=1}^{k} e^{\lambda_v \cdot x}} \tag{27}$$

This is the exact multi-category version of the sigmoid we had to be lucky enough to guess the first time. From this point on you solve as before. This derivation also gives some indication that the sigmoid function (the inverse of the logit link) is in some sense special among the different GLM models (a good indication that you might as well try it until you know something about your problem that indicates you should use some other form).

It might seem that guessing the sigmoid form is less trouble than appealing to maximum entropy. However the sigmoid is special trick (either it is appropriate or it is not) and the maximum entropy principle (and also taking partial derivatives of the Lagrangian) is a general technique. The sigmoid is the "what" the methods we used here (maximum entropy, the Lagrangian and essentially the calculus of variations[Gelfand and Fomin, 1963]) is the "how."

5 Conclusion

We showed how to derive logistic regression twice.

For the first derivation we guessed the useful sigmoid function and then used simple calculus to deriver important properties of logistic regression. This derivation for the sigmoid alone (not introducing general GLMs and not introducing link functions) is much quicker and simpler than the standard derivations (which require some more clever substitutions).

The second derivation we used the central equation of information theory and the Karush-Kuhn-Tucker theorem to actually discover the sigmoid and logistic regression from first principles. This did require brining in knowledge from two different fields (optimization for the Karush-Kuhn-Tucker conditions and information theory for the entropy equation). However in both case we took the most famous fact from a highly celebrated field.

We have worked in detail to show the "how" and called out to larger theories to illustrate the "why."

References

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

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.*

[Gelfand and Fomin, 1963] Gelfand, I. M. and Fomin, S. V. (1963). *Calculus of Variations.* Prentice Hall, Inc.

[Klein and Manning, 2003] Klein, D. and Manning, C. D. (2003). Maxent models, conditional estimation, and optimization.