**Logistic Classification**

Now we’ll consider the case of logistic regression, which tries to fit data to discrete probabilities. So in this scenario, our data points at xi can take on only two values: yi = yes/no, true/false, 1/0. For instance, we might plot ‘Having Insurance’ vs. ‘Age’.

A graph of age and age

Description automatically generated

We could try to create a model to predict whether or not the person will buy insurance based on their age. The first option we might try to fit the data with a linear regression. And we might say the y-value of the line represents probability of buying insurance so that if y > 0.5, we’d conclude they would buy insurance; otherwise no. But this model would be susceptible to outlier issues. For instance, if we have an 85 year old person whose buys insurance, the best fit line above would incorrectly predict people less than about 47 years of age won’t buy insurance. On the other hand, a function like this works better,

A graph of age and age

Description automatically generated

It’s not susceptible to the outlier problem. And it’s called the logistic function, f(x). So we’d try to model data, Yi via,



Take the εi with a grain of salt. We make no assumptions about how εi is distributed, and won’t try to minimize an SSE. In the neighborhood of any given coordinate x, there may be any number of xi’s and associated yi’s equal to 1 or 0. Let the number of yi = 1 values be δn1, and the number of yi = 0 values be δn0. The probability at x of y = 1 would be something like f(x) = δn1/(δn0 + δn1). So locally, the distribution of 1’s and 0’s should be just binomial.

So f(x) looks like a Fermi distribution function reflected horizontally about μ. At x = -∞, it’s 0; at x = μ, it’s ½, and at x = ∞, it’s 1. The β parameter controls the steepness of the curve. Of course at β = 1/T = 1/0, we get a step function centered at x = μ. Can also write f(x) as:



So when is the logistic curve a good fit? It seems that as a general rule, the probability f(x) must asymptote to 0 and 1 at the extremities of the domain. So if yi were shaped more like a sinusoidal function of xi, or wasn’t always monotonically increasing (or decreasing if we flip the curve around) this would not be a good regression curve. And also, the y = 1’s would have to be more or less separable from the y = 0’s, at some point in space.

Apparently the parameters of the best fit curve are determined by maximizing the probability that the curve correctly predicts the values of all data points. So let our points be xi and their binary values be Yi, where Yi can be 0 or 1. Then let f(xi) be the value of the logistic regression curve at the point xi. Then we’ll define the cumulative likelihood as:



To clarify, if Yi = 1, then the probability we’d classify it as such is f(xi). And if Yi = 0, then the probability we’d classify it as such is 1-f(xi). So this is the probability that we classify everything correctly. So it looks like when we maximize Lf, we’re maximizing the model’s Accuracy = (# correct predictions/# predictions) (see Evaluating ML Models file)? Actually, I don’t think it is the same. Can also write this as:



where i = 1, 2, …, n0 enumerates the points where Yi = 0, and j = 1, 2, …, n1 enumerates the points where Yj = 1. And we’d try to maximize this likelihood. Alternatively we could define the log-likelihood, ln(Lf), and maximize *that*. Or…we could construct the information ‘log-loss’ by taking the -ln of Lf. and minimize *it*. I imagine it’s called the information loss because by approximating all the discrete points with an interpolating function, we’re losing information. Well the log-loss LLf is:



Can also write as:



And, can also write as:



where that sum extends over all n points. And one more nice manipulation. Let’s call f(xi) = f1(xi), the calculated probability of a yes outcome. And we can say 1 - f(xi) = f2(xi) is the probability of a no outcome. And let’s write Yi = p1(xi), the probability of a yes outcome, and then 1 – Yi = p2(xi) will be the probability of a no outcome. p1(xi) and p2(xi) are either 1 or 0 of course. In this language we can say,



Either way, we’d take derivatives w/r to β and μ and set to zero I guess. Doesn’t look like we can find an exact solution. So probably need to do a gradient descent routine to numerically solve for β, μ.

Looks like we can take LLf to be analogous to the SSEf from linear models. It will be zero when there is perfect fit, and ∞ for really bad fit.

By the way, fitting a logistic regression curve to minimize LL looks like this:

Chart, line chart

Description automatically generated

and if we minimize SSE instead, we get this:

Chart, line chart

Description automatically generated

So they’re kind of close. The LL guy hugs the 0’s and 1’s a little more closely. And overall, it does kind of seem like a better ‘fit’.

**Example:**

If all the yes’s are separated from all the no’s by some space Δμ, then wouldn’t any curve with β = ∞, and μ in between Δμ work? I guess we can have multiple solutions then. But yeah such a curve would go straight through all the data points, or arbitrarily close to them all. And so we’d have:



**Example:**

In the other extreme, say that for every data point (xi, yi = 0), we have another (xi, yi = 1). Rather say that at every point xi we have n1 yi = 1’s and n0 yi = 0’s. Then would we simply get f(xi) = n1/(n0 + n1)? I’m going to write,



and I’m going to bias my analysis by presuming from the start that m = 0. Then f(x) = p, some constant. Now let’s form LLf,



Now differentiate w/r to p and set to 0,



So that works out.

**Example**

Let’s generalize the last example a little bit. Say we just have n1 yes’s and n0 no’s scattered about at whatever xi’s. And then let’s say we fit the data with a logistic regression curve f(x) = p again. What will p be? Well we form the log-loss:



Now differentiate w/r to p and set to 0,



So same as before. Parenthetically, the distinction between them is that I’m claiming our solution for the specified scenario in the previous example is exact. In this example it is not exact – just the best we can do with the model f(x) = 1/(1+e-(0·x+b)) = p. It stands to reason that any logistic function which retains the m d.o.f. will do a better job. So what is LLf in this case?



Okay.

**Goodness of Fit: R2 value**

R2 is a measure of the goodness of fit, of the regression curve. There’s apparently no consensus on a definition of R2. But one analogous to the linear models expression is this. We define



where LLf is the log-loss of the logistic regression curve, and LLm is the log-loss of the curve that just fits the ‘mean’ of the data, i.e., it is f(x) = p = n1/(n0 + n1), where n1 is the total number of yi = 1’s and n0 = the total number of yi = 0’s. As we argued in the example above, LLm forms the upper bound of the log-loss for any logistic function. So we should always have LLf < LLm. And so 0 < R2 < 1.

**Hypothesis Testing**

Say you find a trend in the data, via all the regression stuff above. But someone else says that the data is actually just randomly 1’s and 0’s, and there really is no trend. How could we test this hypothesis? Basic way is this. Let’s define a Null Hypothesis.

H0 = assumption that the data is described by model f(x) = p = n1/(n0 + n1)

And note LLm be the error associated with this.And might also note that this is just a logistic regression model with m = 0. Then let’s compare to our full model with f = 2 degrees of freedom, fitting f(x) = 1/(1+e-β(x-μ)). And let SSEf be its log-loss. We would anticipate this to be smaller of course, i.e., SSEm > SSEf. The alternative hypothesis would be:

HA = assumption that the m parameter in the new model f(x) = 1/(1+e-β(x-μ)) is non-zero.

Well we can form a test statistic,



Turns out this follows an χ2-distribution with 1 d.o.f. (the difference in degrees of freedom of the f model (2 d.o.f.) and the m model (1 d.o.f.)).



which is the probability density of getting an Z-value of x, given the null hypothesis is true. So we can calculate a p-value,



which would be the probability that we’d get an Z-statistic value of Z\* or higher, out of the new model, if the Null hypothesis were true. So if the f model has true explanatory power, then we should find Z\* >> 0 and the p-value should be small (less than 0.05 at the 95% significance level).

**Comparison to other Models**

We can try to use the logistic regression fit to classify a new test point, **x**, as either 0 or 1. To do so, we’d usually make some cut-off like p = 0.5. And if f(**x**)>0.5, we’ll classify it as 1; otherwise 0. The logistic regression method might work better than say KNN if, as above, there is no sharp well defined boundary separating the two cases, and there are a lot of ‘outliers’. Looks like it would also require the 0’s and 1’s to be roughly separated by a d-1 dimensinal hyperplane.

**Appendix 1**

Note the log-odds of f(x) is defined as:



This maps the probabilities, confined between (0,1) on the y-axis, to a range (-∞,∞).

**Appendix 2**

Since f(x) → probabilities. Can we say f(x) = P(y|x)? Yes I think so. Then what is P(x,y)? Well,



and I guess P(x = xi) would just be 1/n, where n = # data points, presuming there are no duplicate xi’s. So then we could say,



and normalization requires,



Okay.