

Using log-transform to avoid underflow problem in computing posterior probabilities

Yang Wen

Last updated: September 16, 2007

Abstract

This article presented a log-transform approach to avoid underflow problems, which could lead to a difficult to ignore division-by-zero problem, in computing posterior probabilities under the latent-class model context. We analyzed the characteristics of the problem, gave a solution algorithm, and discussed some of the practical issues. The technique we used can be easily adapted to other contexts.

1 Context

Recently I worked on some tasks that used latent-class models. In particular, we had a sample dataset of many users, each with multiple observations. We would like to classify the users into several cohorts (latent-classes).

We had a set of covariate variables that we thought would be relevant, and assuming we could somehow estimate the coefficients for those covariates, we could easily compute the prior membership probability for each user, i.e., the prior probability of that user belongs to a given class.

For several reasons, those priors might not be accurate. However, since we observed each users for many times (with different outcomes), it would be possible to improve our membership probability estimation by taking those information into account, i.e., computing the posterior membership probabilities.

We could potentially process huge amount of data, so all those processing had to be done by computers. While testing our code, we encountered an interesting underflow (or overflow, details later) problem. This article describes what the problem is and how we deal with it accurately and efficiently.

2 How to compute the posterior membership probabilities

We first introduce our notation. For simplicity we assume the model in question is a rating model. (For choice models the equations need slight changes but the ideas are the same.)

Generally we denote y as the dependent variable. For a rating model, the users face a problem of whether or not to pick some items (e.g., put a DVD into their wish list or shopping list). If the item is chosen, then the dependent variable is considered as 1, otherwise, 0.

Let $Y_n = (y_{n1}, \dots, y_{nT_n})$ be the vector of observed values of the dependent variable for user (decision-maker) n . The number of observations we have for user n is T_n .

Let X be the set of attributes for the items (and possibly for the given user). Let S be the max number of possible cohorts. For each cohort s ($s = 1, \dots, S$), our model says the probability of an item being chosen depends on the X and a set of parameters β_s , i.e., our model gives $P(y|X, \beta_s)$.

Meanwhile, for each user n , we have a set of covariate variables (characteristics of the user) Z_n , and correspondingly a set of parameters θ , from which our membership model can compute the prior probability $Q(s|Z_n, \theta)$, for each $s = 1, \dots, S$.

For single choice problem, we have the ‘‘expectation’’ of an item being chosen given as

$$P(y|X, Z_n) = \sum_{s=1}^S [P(y|X, \beta_s)Q(s|Z_n, \theta)]$$

Given we have multiple observations for each user, the above equation should be written as

$$P(Y_n|X, Z_n) = \sum_{s=1}^S [P(Y_n|X, \beta_s)Q(s|Z_n, \theta)] \quad (1)$$

If we assume local independence, namely $P(Y_n|X, \beta_s) = \prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_s)$, then Equation (1) becomes

$$P(Y_n|X, Z_n) = \sum_{s=1}^S \left\{ \left[\prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_s) \right] Q(s|Z_n, \theta) \right\} \quad (2)$$

Using the Bayesian formular, the posterior membership probability is written as

$$Q(s|Z_n, \theta, Y_n) = \frac{P(Y_n|X, \beta_s) Q(s|Z_n, \theta)}{\sum_{j=1}^S [P(Y_n|X, \beta_j)Q(j|Z_n, \theta)]} \quad (3)$$

$$Q(s|Z_n, \theta, Y_n) = \frac{[\prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_s)] Q(s|Z_n, \theta)}{\sum_{j=1}^S \{ [\prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_j)] Q(j|Z_n, \theta) \}} \quad (4)$$

Equation (4) is obtained from Equation (3) by assuming local independence, similar to Equation (2).

3 Underflow problem in computing the posterior membership probabilities

There are quite a few place we can face potential overflow or under flow problems. For example, in computing the prior membership probabilities ($Q(s|Z_n, \theta)$, $s = 1, \dots, S$), we generally use the Logit formular, in which we specify an utility function and compute the probabilities by taking the exponential transform of the utilities, and exponential transform

could lead to an overflow problem. The solutions to this type of problems have been well-discussed in previous literatures. One often tries to normalize the range of the variables properly before the computation. For this article we assume this type of problem does not occur.

There is another type of overflow or underflow problem, which occurs less often, but will cause problem for computing the posterior membership probabilities.

As one can see from Equation (4), to compute the posterior membership probabilities, we need to multiply $P(y_{tn}|X_{tn}, \beta_s)$, for all $t = 1, \dots, T_n$. Remember that T_n could be a big number (as we could easily have hundreds of observations for a single user), and in general we have $0 < P(y_{tn}|X_{tn}, \beta_s) < 1$ (since it is unusual to have the probability equal to 1 or 0). Then for some users (especially those for whom the model does not predicting well), $\prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_s)$ could be a very small but positive number.

There is a limit, however, how small a floating point number can be accurately represented on each computer system, if it uses a fixed number of bits to represent a floating point number. For example, on a typical 32-bit PC, according to the commonly used IEEE 754 standard format, a “double precision” floating point number occupies 64 bits (8 bytes), among which 11 bits are used for the exponent. The positive number closest to zero would be somewhere around 10^{-324} . If we have a smaller (but positive) number, in the computer representation it becomes zero. Such situations are referred to as “underflow” problems (or “overflow” in the exponents).

In practice, the products in both the numerator and the denominator of Equation (4) can be so small that they cannot be represented by the smallest positive floating point number in the computer system, and then the “underflow” problem occurs. If the underflow occurs only in the numerator, then our posterior membership probability given by Equation (4) will be an unrepresentable value and, in most cases, it can be safely ignored and zero substituted. On the other hand, if the underflow occurs in the denominator (in which case it should also occur in the numerator), then it might lead to a later division by zero error which cannot be so easily ignored.

4 Using log-transform to avoid the underflow problem

Whether or not an underflow problem occurs, the posterior membership probabilities $Q(s|Z_n, \theta, Y_n)$ (for $s = 1, \dots, S$) should not be all zero, because they should add up to one:

$$\sum_{s=1}^S Q(s|Z_n, \theta, Y_n) = 1$$

When an underflow problem leading to a division by zero error occurs, it means for all s , $\prod_{t=1}^{T_n} P(y_{tn}|X_{tn}, \beta_s)$ is very small. Theoretically, we can multiply both the numerator and the denominator by a suitable number, then at least for some s' , both the numerator and the denominator should be within representable range of the floating point numbers, and the underflow should not happen. While such an idea sounds straightforward, the difficulty is, however, that it is hard to choose such a multiplier that is guaranteed to work without introducing other problems. For example, since the choice of multiplier affects all cohorts, one needs to process all observations for a given user across all cohorts and then find out

the suitable number, which is user-specific. In other words, we need to do quite some bookkeeping work for each user to make sure the multiplier work in all extreme cases.

A simple to implement but less efficient approach is to use the log transform on $P(y_{tn}|X_{tn}, \beta_s)$. Then instead of multiplying $P(y_{tn}|X_{tn}, \beta_s)$ together, we can sum up $\ln P(y_{tn}|X_{tn}, \beta_s)$, which is quite unlikely to be overflow, for most practical purpose.

Let

$$R_s = \ln P(Y_n|X, \beta_s) = \text{sum}_{t=1}^{T_n} \ln P(y_{tn}|X_{tn}, \beta_s), \quad \forall s = 1, \dots, S \quad (5)$$

$$K = \max_s R_s, \quad (6)$$

$$D_s = R_s - K, \quad \forall s = 1, \dots, S \quad (7)$$

Then we have

$$e^{D_s} = e^{R_s}/e^K = P(Y_n|X, \beta_s)/e^K, \quad \forall s = 1, \dots, S$$

$$P(Y_n|X, \beta_s) = e^{D_s} * e^K, \quad \forall s = 1, \dots, S \quad (8)$$

Substitute Equation (8) into Equation (3), we get

$$\begin{aligned} Q(s|Z_n, \theta, Y_n) &= \frac{e^{D_s} * e^K * Q(s|Z_n, \theta)}{\sum_{j=1}^S [e^{D_j} * e^K * Q(j|Z_n, \theta)]} \\ &= \frac{e^{D_s} Q(s|Z_n, \theta)}{\sum_{j=1}^S [e^{D_j} Q(j|Z_n, \theta)]} \end{aligned} \quad (9)$$

Note that the denominator in Equation (9) will no longer be zero, because $\max D_s = \max R_s - K = 0$, $e^0 = 1$, and $Q(s|Z_n, \theta) > 0$ (The priors are strictly positive if a Logit formula is used and no overflow occurs in the exponential transform by our assumption).

Therefore, we can compute D_s (for all s) from Equation (7) and use Equation (9) to get the posterior membership probabilities, without worrying about the underflow problems.

One good thing about this approach is that the processing for each cohort is independent from other cohorts. There is no tedious bookkeeping across cohorts.

5 Practical issues

Efficiency consideration Generally speaking, taking log-transform is (considerably) expensive than multiplication or additions. Therefore this approach should be used as a backup, and only used for users for whom the underflow problem causes the denominator in Equation (4) to be zero.

Applicability We have implemented this approach in Stata and tested it for several large datasets. One might point out that this log-transform approach will still be broken if, in Equation (5), $\text{sum}_{t=1}^{T_n} \ln P(y_{tn}|X_{tn}, \beta_s)$ causes overflow problems. That is true. But we would argue that, compared with the original case in Equation (4), the possibility of overflowing in Equation (5) is almost negligible. In general, $0 < P(y_{tn}|X_{tn}, \beta_s) < 1$, the sum must be negative. The only way it gets overflowed is when this sum goes smaller than the smallest floating point number (which is around -10^{308} for the usual 64-bit double precision representation).

6 References

- http://en.wikipedia.org/wiki/Floating_point
- http://en.wikipedia.org/wiki/IEEE_754