Semi-Byzantine Reputation Networks Using Graphical Models

Daniel Speyer
December 19, 2016

Abstract

Ratings systems are used in many areas of life, but most treat dishonesty and confusion as special cases, handled by heuristic, ad-hoc input filtering. The exceptions are fully byzantine systems, which tend to treat trustworthiness as binary and seek provable guarantees. Neither is a good fit for real world problems, in which both fuzziness and dishonesty are ubiquitous, and probabilistic conclusions are the best we can hope for. In this paper, I will introduce a graphical model that can handle some of these problems, demonstrate where it does and does not work, and discuss what would be necessary to make it applicable in practice.

1 Introduction

Establishing trust is a necessary precondition to many profitable exchanges and collaborations. This is particularly difficult between private individuals without social links. Large organizations have invested a great deal in their names and therefore have something to lose. Individuals with social links have plenty of evidence and leverage. But there are cases in which people want to establish trust without these resources. For example:

- An individual-to-individual auction site, as eBay was originally intended
- Any form of sharing economy
- Non-monetary reciprocal room sharing (as it common among dancers)
- Coordinating refugees fleeing an oppressive government

The natural solution is some sort of reputation system. Primitive rating systems such as Yelp have provided much value. Nevertheless, the system is vulnerable to varying standards, to mistakes, and to dishonesty.

2 A Simplified Model

To investigate, I made two simplifications. I collapsed all attributes of interest into a single "goodness", and all factors that could cause a rating to be wrong into a single factor called "folly" (this name is a reminder that a high value indicates a less reliable rating).

If a rating exists for a rater/ratee pair, it contains two values: an estimate of goodness ("rating") and an estimate of folly ("meta-rating"). Ratings are *expected* to be sparse, represented by a "has_rating" boolean flag (since a rating of 0,0 is *very* different from no rat-

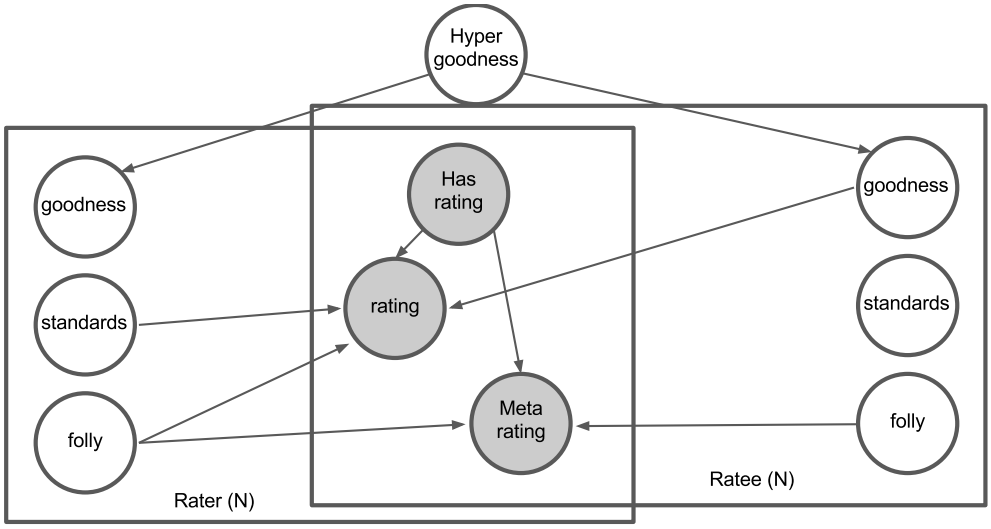


Figure 1: The simplified graphical model. The rater and ratee are shown separately for clarity, but they are drawn from the same pool of users.

ing at all).

In addition to goodness and folly, each actor has an attribute called "standards". This represents the fact that there is no objective units with which to measure what the users are asked to measure. Rather than try to communicate a set of standards and ask users to apply them, one can present simple sliders from "very good" to "very bad" and learn post-hoc what those mean.

It is now possible to describe the model mathematically:

$$R_{i,j} \sim \begin{cases} \mathcal{N}(G_i - S_j, F_j) & HR_{i,j} \\ NaN & else \end{cases}$$

$$MR_{i,j} \sim \begin{cases} \mathcal{N}(F_i, F_j) & HR_{i,j} \\ NaN & else \end{cases}$$

Note that this model has *one* too many degrees of freedom: *all* goodness and standards values could be increased by the same constant without effecting the distributions for the observations. This is solved by declaring that the standards are drawn from a normal distribution centered on zero. There is *no* similar assumption about goodness, instead it is treated as drawn from a distribution whose parameters can be learned. The overall goodness of a community, and therefore what to expect from an unrated user, is very much something worth learning.

For thoroughness sake, folly is drawn from an inverse gamma distribution and the goodness hyperparameters from an improper flat distribution. The "has rating" observation is not drawn from anything, so nothing is learned directly from it.

Note that I have now introduced three arbitrary constants: the variance of standards and the two parameters of the folly distribution. A more sophisticated model might try to learn them, or

to pick them intelligently. For now, let us stick with constants that make testing easy.

2.1 Multi-Attribute Models

A real-world version of this would have multiple attributes, instead of a single "goodness". For example, an apartment sharing network might have "reliability about making contact" and "apartment is clean". This is *necessary* to make the model work. If any question could be answered "well, it depends what you care about," then that question *must* be divided. Otherwise there will be variation in ratings for reasons the model cannot describe, resulting in confused and useless learned values.

Dividing goodness into attributes is very domain-specific, so I will not attempt it here.

It might be tempting to use matrix factorization to find the attributes, but this would produce a less useful result. Users will often want to know not just "Will I be happy interacting with this person?" but "What will I be happy or unhappy about?". The latter allows one to plan around a problem.

Multiple attributes allow a few extensions to the model. One could introduce a general factor of high standards, or a horns/halo bias parameter (horns/halo bias is the tendency of people to think that all good traits about another person are more correlated than they really are). But these extensions are not very interesting, so nothing important is missed by omitting them from the testing model.

2.2 Separating Confusion and Dishonesty

In the simplified model, I use a single factor to describe being wrong. This misses important behavior. A large number of very unreliable people having the same opinion is still significant evidence, but a large number of dishonest people is not. However, this does not effect the interesting theoretical properties of the system, so I omit it here.

2.3 Other Extensions

There are many additions one might make to this model, such as self-reported confidences, outside information sources, or an concept of time. Having a model like this makes it very easy. You just include them in the distribution of the ratings. Testing becomes more difficult, but applying the model with Hamiltonian Monte Carlo does not. This flexibility is a big advantage of the graphical approach.

3 Testing

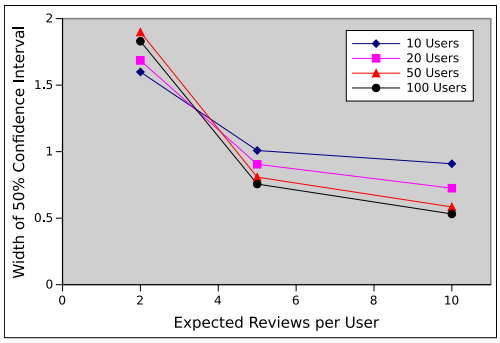
Lacking real-world training data, I tested the model on multiple simulations.

3.1 The Simplest Case

The simplest case is where the model is correct. I used Stan to generate data from the model, then stashed away everything except the ratings, deleted most of those, and used Stan again to generate the missing parameters.

This was subject to one systematic flaw. If the average of the generated standards was not zero, all ratings were off by that amount. The standards were drawn from a normal distribution with zero mean, but the sample size was small. Since this would go away in a larger test, and is an uninteresting sort of failure to begin with, I adjusted all values to remove it.

I then set all the follies to 1 to make the results more readable. I looked at the 50% confidence intervals for all the attributes, which contained the (adjusted) true values between 43% and 60% of the time.



With only five expected ratings per user, the interval was close to the folly, and it shrank with more. More encouragingly, a larger userbase consistently gave better results. (I refer to "expected" ratings count because I selected whether each pair had ratings independently with constant odds.)

3.2 Internal Variance

Suppose there is a user who receives bimodal ratings from users who have rated *other* users closely (and so do not have wildly differing standards). The posterior will peak in between the modes with a narrow variance. This is not what we should conclude. Nor should we conclude a bimodal posterior. What we should conclude is that the model itself is wrong, and there is no single goodness value to approximate. Perhaps the ratee is inconsistent, or the attributes are not adequately divided.

This case can be detected by looking at the log probabilities of the ratings. We do not expect these to be zero. In fact, the expected log probability is the entropy, which is known to be $\frac{1}{2}\log(2\sigma^2\pi e)$. If the actual log probability is consistently more extreme than expected, something is amiss.

For each rating, divide the actual negative log probability by the entropy to get a sort of "absolute surprise" rating. If the average of this for all ratings a ratee has exceeds a constant (three, in my tests) throw out the result altogether. One could devise a more elegant solution by finding the probability-of-probability curve (though this is complicated for a continuous distribution), but it would still end in an arbitrary constant, so I might as well stick with this.

This method consistently found the test examples I wrote, with no false positives in 10 runs of 20 users generated from the model.

3.3 Symmetric Opposing Cliques

Suppose we have one controversial person and two equal size groups giving him opposite ratings. Furthermore, everyone within each group vouches for their groupmates' judgment and denigrates the opposing group's. In this case, there are two equally good posteriors. Simply running the HMC 4 times with random seeds and checking for nonoverlapping 50% intervals is sufficient to detect this.

Again, this consistently found my test cases with no false positives in 10 runs of 20.

3.4 Asymmetric Opposing Cliques

However, if one clique is larger than the other, there is a single best fit posterior. Furthermore, HMC will find it. Even if the cliques are sizes 5 and 6, and I initialize to believing the 5 and not the 6, HMC will consistently find the global maximum.

There is still a local maximum at believing the smaller clique. If the algorithm could find it, it could compare the log probabilities and produce a bimodal final posterior. Unfortunately, neither extreme initialization values nor low iteration counts suffice.

Somewhere there may exist an optimization algorithm specifically designed *to* get stuck in local optima, but I haven't found it. And if I had one, I would not be able to usefully validate it.

4 Conclusion

Graphical model techniques do an excellent job of handling differing standards and ordinary failures of judgment. They can detect when they have failed, but doing so involves unvalidated thresholds.

Handling multiple optima remains an open problem. Successful handling of the perfectly symmetric case is not very valuable, as perfection is not found in nature, and the support here is brittle.

The validation problem is particularly distressing. Even if we had an optimization algorithm that found all the optima in our test cases, how could we establish that it did with real data? After seeing the complexity of the cases, I can think of no solution that does not involve plentiful, real-world training data.

So, even though graphical models look promising, and the algorithmic problem looks solvable, the "validate in theory and simulation, then apply in the real world" methodology is not looking promising. Given the dearth of reliable training data, this may be a death knell for the entire approach.

At least I have achieved useful insight into how difficult the problem is.