Problem Solving*

Gaurav Sood†

December 30, 2023

The first rule of problem-solving is to diagnose before looking for solutions. Though like all rules, the exception proves it. Sometimes it is cheaper to try out solutions to diagnose and solve a problem. For instance, to diagnose allergies, you could give an antihistamine. If the symptoms improve, the likely cause of the runny nose is an allergy.

1 Diagnosis

There are three parts to diagnosis: 1. Generating explanations, 2. Generating priors for the explanations, and 3. Isolating the cause.

1.1 Generating Explanations

There is an art to generating explanations. Crude explanations are rarely helpful. For instance, one could posit that the model is failing because the data is 'bad'. However, you need to articulate what is wrong with the data, e.g., the camera has one or more dead pixels, for the discovery to be actionable. One way to get to granular hypotheses is to pursue the three why methods—also called the five-year-old's method—ask why enough times to get to a precise enough hypothesis.

There are three broad ways of generating explanations:

- **Finding correlations.** To generate the candidate set, find variables correlated with the error. Say, for instance, that you are looking to explain why the ETA prediction model is failing. To find explanations, test if the error varies by location, time of day, day of week, etc. It is often useful to test correlations with a continuous rather than a Boolean variable. This allows you to test if an increase in the source variable causes an increase in the effect. For instance, say you wanted to ascertain if occlusion is causing worse performance, you could check to see if it is true that the greater the occlusion of say the traffic light, the worse the performance.

*The article benefited from comments by Rohit Pal and Daniel Stone.

†gsood07@gmail.com.

- **Learning from failures.** Selecting on the dependent variable is rightly frowned upon. When you select on the dependent variable, even correlation is not guaranteed. However, analyzing failures is the go-to trick for generating explanations. It can be thought of as the inductive reasoning method. We use an example (or examples) to develop a more general hypothesis. The process may work as follows: Start by selecting failures. Sample failures randomly. Or pick the worst errors; the worst errors are often the site of the most obvious problems. Next, look at each of these examples closely. For instance, you may want to trace the example through the system. Or you may want to compare the failure with 'similar' successful cases to find potential patterns. Say you run an ETA prediction company. Say when you look at the data with the worst misses, you discover that the locations you are getting a minute apart are hundreds of miles apart. This can then lead you to the diagnosis that your application is installed on multiple devices.

- **Ideation.** The first two methods build on existing data but existing data may not be enough. Often, it is useful to ideate independently about potential causes of failures and then investigate if you have the right data to triage the problem. Plausibly the most important way to look at the problem is from a systems perspective. Use it to figure out where problems can happen.

1.2 Generating Priors

The commonest path to priors is prior experience. Priors can be used in a priority matrix to investigate which explanations should be investigated first. A fuller prioritization matrix for such an exercise may include plausibility, impact, cost-effectiveness, speed of investigation (but plausibly also the speed with which we can implement a solution), whether or not the cause is within the span of control, and ease of implementation of solutions.

1.3 Isolating the Cause

Isolating causes from a proposed set of causes has a rich intellectual history going back at least to Mill's methods. But before we go to specific techniques, there is a meta-thinking tool: Mutually Exclusive and Comprehensively Exhaustive (MECE). Relentlessly working to pare down the problem into independent parts is among the most important tricks of the trade. Let's see it in action. After looking at the data, an engineer finds that couriers arriving late is a big problem. But why are the couriers not arriving on time? It could be because of 'bad' couriers or it could be that couriers are being set up for failure. These mutually exclusive and comprehensively exhaustive parts can be broken down further. You may be setting couriers up to fail by giving them too little lead time or by not providing precise directions. If you go down yet another layer, the short lead time may be a result of you taking too long to start looking for a courier or because it takes a long time to find the right courier. So on and so forth. There is no magic to this. There is no science to it either. MECE tells you what to do but not how to do it. It simply requires iterative ideation.

There are five techniques for isolating causes:

- **Similar Others.** Compare to similar others. We can even generate similar others. For instance, we can use GANs or generative models to edit the scene except for one feature (Athey et al. 2022).

- **Dr. House.** The good doctor was a big believer in differential diagnosis. Dr. House often eliminated potential options by evaluating how patients responded to different treatment regimens. For instance, he would put people on an antibiotic course to eliminate infection as an option. The more general strategy is experimentation: learn by doing something. In ML, we can A/B test systems.

  Experimentation is a sine-qua-non where people are involved. The impact of code is easy to simulate (though it is harder to simulate the impact of data as a way to root cause data problems). But we cannot answer how much paying $10 per on-time delivery will increase on-time delivery. We need to have a controlled experiment.

- **Which parts of the system are working?** The error in complex machine learning deployments can stem from lots of sources—hardware failures, e.g., lighting failure, camera error, server error, network error, data errors, algorithm failure, etc. Keep metrics for each major component that is essential to prediction. For instance, for servers, we track latency and throughput. This allows us to eliminate potential explanations. Funnels and flow charts or process diagrams are two ways to think about problems.

  One way to figure out where the problem is is to exploit time. Start with 100% and draw the Sankey diagram, popularized by Minard's Napoleon Goes to Russia. Funnels are powerful tools capturing two important aspects: how much we lose in each step, and where the losses come from. There is, however, one limitation of funnels—the need for categorical variables. When you have continuous variables, you need to decide smartly about how to discretize them. Following the example we have been using, the heads-up we give to our couriers to pick up something and deliver it to the customer is one such continuous variable. Rather than breaking it into arbitrary granular chunks, it is better to plot how lateness varies by lead time and then categorize at places where the slope changes dramatically.

  There are three things to watch out for when building and using funnels. The first is that funnels treat correlation as causation. The second is Simpson's paradox which deals with issues of aggregation in observational data. And the third is how the coarseness of the funnel can lead to mistaken inferences.

  An analog to the funnel is a system diagram (and ideally a state machine for all finite state machines). It generally pays to know how the cookie is baked. Learn how data flows through the system and what decisions we make at what point with what data and what assumptions for what purpose. Conventional tools are flow charts and process tracing. For example, an AV error may stem from failures in the perception system. The errors in the perception system may be because of bad inputs, e.g. if the input sensor(camera) has a dead pixel in the middle of its acquisition region, and if you rely on a simplistic color-based traffic light detector model, you may suddenly see a large increase in errors.

Or errors may stem from incorrect logic. For instance, if you don't capture the variance of illumination based on time and effective lux measurements based on weather, you may detect a single class. Errors can also stem from integration-level issues between two subsystems.

- **Physicist's Method.** Find the explanation that fits *all* the data. Generally, that explanation is unique (see the section on Maxims).

1.4 Maxims

The five maxims for diagnosis are:

1. **Too good (bad) to be true.** It pays to have a mental model of what to expect. If something looks like it is too good (bad), it is likely wrong. This is called Twyman's law: "The more unusual or interesting the data, the more likely they are to have been the result of an error of one kind or another."

2. **Single point of failure.** Generally, there is only one thing wrong rather than a few different things.

3. **Skew.** Opportunity is often concentrated in a few places. Pursuing our example above, it could be that a small proportion of our couriers account for most late deliveries. Or it could be that a small number of incorrect addresses cause most of the late deliveries.

4. **Obvious is underrated.** Odds are that the problem is obvious. For example, one study of large Internet services found that configuration errors by operators were the leading cause of outages, whereas hardware faults (servers or network) played a role in only 10–25% of outages (Kleppmann 2017).

   If things look too complicated, there is a high chance that your metrics or mental model are wrong.

5. **Fixing issues upstream is generally better than fixing downstream issues.** For instance, we can invest in building models that surmount bad data but it is generally cheaper to fix the data.

2 Solution

There are three broad steps to getting to a solution: Start by generating a list of solutions. Next, select between the options based on some kind of priority matrix. (Sunstein and Hastie (2015) argues it is optimal to split discussions on the generation of options from discussions on selection between options.) Third, expand on and implement the solution. Often you need to iterate over the solution as the initial solution doesn't work. And when it happens, there are natural questions about whether the diagnosis was wrong or whether the solution was bad. Often, it is both. Problem-solving as part of continuous improvement is rarely a one-shot.

To illustrate the process, let's return to the example of late-arriving couriers. Let's say that the primary reason for late deliveries is couriers, and not systemic factors like lead time, poor directions, and such. Say the broad solution we have come up with is to incentivize on-time delivery. But how? We could build past performance into what bids are accepted from which couriers. But that may not be enough. We may need to inform couriers about the potential implications of the new policy. But when designing this policy, we may want to leverage behavior economics, e.g., loss aversion, in how we inform couriers.

References

Athey, Susan, Dean Karlan, Emil Palikot and Yuan Yuan. 2022. Smiles in profiles: Improving fairness and efficiency using estimates of user preferences in online marketplaces. Technical report National Bureau of Economic Research.

Kleppmann, Martin. 2017. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. " O'Reilly Media, Inc.".

Sunstein, Cass R and Reid Hastie. 2015. *Wiser: Getting beyond groupthink to make groups smarter*. Harvard Business Press.