Grab Bag 2: More Frequently-Asked Questions (and Answers) About Data Mining

By Tim Graettinger

For the past year, I have presented a data mining "nuts and bolts" session during a monthly webinar[1]. My favorite part is the question-and-answer portion at the end. In a previous article[2], you learned my thoughts on: "*what tools do you recommend?*", "*how do you get buy-in from management?*", and "*how do you transform non-numeric data?*" Since my cup overfloweth with challenging, real-world questions from the webinar, it's time for a sequel. This time, we'll focus on data and modeling issues. Let's get to the questions.

Question 1: How much data do I need for data mining?

This is by far the most common question people have about data mining (DM), and it's worth asking why this question gets so much attention. I think it's almost a knee-jerk response when you first encounter data mining. You have data, and you want to know if you have enough to do anything useful with it from a DM perspective. But despite the apparent simplicity of the question, it is unwise to try to answer without digging deeper and asking yet more questions. My goal here is to provide you with the guiding principles you need understand so you can ask those next questions. You'll even get a rule of thumb so you can produce your own estimate of the data you'll need for DM.

One guiding principle is based on **relationship complexity**, that is, the complexity of the relationship you want to model. The more complex the relationship, the more data you need to model it accurately. Duh, right? But, ask yourself, "What's the problem with this guiding principle?" Did you say, "I don't know how complex the relationship is?" Good. From a practical perspective, it's useful to think of complexity in terms of the number of factors that might play a role in the relationship. Let's say that you want to predict customer churn. Think about the probable factors that might impact churn, such as: tenure, age of the customer, number of complaints, and total lifetime value of purchases,

among others. Are there 4 probable factors or 14? Don't be concerned about fine precision here. You just want to get in the right ballpark.

With your factor estimate in hand, think next about what you would do to collect data from an experiment involving those factors[3]. Have you thought about it? At the very least, you want to test high and low values for each factor – independently, so you can see their effects without any confounding. And, you want to run each experiment multiple times, to reduce the impact of noise or other spurious events. We can translate these considerations into a handy rule-of-thumb formula:

$$NR \geq M \times 2^{(F+1)}$$

Where F is the number of factors, M is the multiple for each experiment (25 is a useful value here), 2 represents the need for at least high and low values, and NR is the result – the minimum number of records you will want/need for data mining.

Let's do a quick example. Suppose you identified 9 factors for your application. Using the rule of thumb with a multiplier (M) of 25, we get

$$NR \geq \times 2^{(9+1)} \approx 25{,}000$$

which is fairly typical. Notice that, according to our rule, the data requirements for DM rise rapidly with the number of factors – and even become astronomical for 50 or more factors. People earnestly tell me that their application has 50, 100, or even more factors. My response is that not all of those factors occur, or can be varied, independently. And that's what really matters for our rule to be applied. If you think you have a LOT of factors, just use F=12 or 13 in the rule as a good place to start.

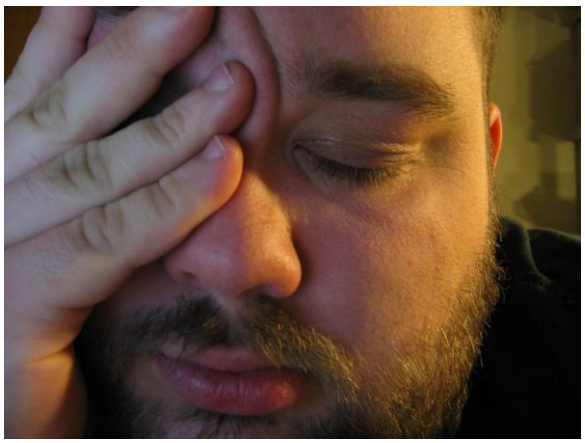
A second guiding principle is **balance**, especially, balance in terms of the various outcomes. In the customer churn application, there are two outcomes: defect and renew. When building DM models, you need data associated with all of the outcomes of interest. The more outcomes you have, the more data you need.

But not only that, you need an adequate mix of each outcome. Are you asking, "What makes an adequate mix?" I hope so. To make our discussion concrete, let's work with the two outcomes for customer churn: defect and renew. Suppose you have 100,000 customer records, but just 1% of them are defections. In other words, only 1000 defections are included in the data set. The number of records associated with the least-frequent outcome becomes the limiting constraint. 1000 records sounds like a small amount (compared to 100,000), doesn't it? In our rule-of-thumb formula above, NR really refers to the number of records associated with the least-frequent outcome. Think about why[4].

A third guiding principle is **model complexity**. The more complex the model you choose to build (in terms of parameters/coefficients), the more data you need.

Again, duh - but there is more to this principle than might be apparent on the surface, and we discuss the details in the context of the next question, posed below …

Question 2: My model performs well, even great, on my training data. However, the performance seems almost random when I test it on new data. Arrghhh! Help!!!



What you're seeing is a classic symptom of "overfitting", or "memorizing", the training data. I've seen overfitting most often in my own work when building models around rare events – like engine failure for a predictive maintenance application or major gift-giving (say, gifts of $10,000 and up) for non-profit fund raising.

When the outcome of interest is rare (like 1 in 100, 1 in 1000, or even less), there is a tendency to build models that are very complex - too complex given the available data. Complex models have lots of free parameters, or handles, which can be adjusted to predict, or "fit", the training data. The problem is that the model conforms to the unique features of the training data so well that it doesn't work anywhere else. It's like custom-tailoring a suit to your unique body. It fits you perfectly, but it doesn't fit anyone else. In the world of model-building, that's no good. We want to build models that are robust, that generalize well to new data.

What can you do?

- First, make sure you create separate training, testing, and validation datasets. That way, you can actually detect a problem when it exists.

- Second, calculate the fitting ratio for your model-data combination. For your model, count or estimate the number of parameters/coefficients. For example, a logistic regression with 10 input elements has 11 coefficients[5]. Alternatively, a backpropagation neural network with 10 input elements and 5 hidden-layer nodes has 61 coefficients[6]. Then, for your data, count the number of records from the minority class (the "rare" event, e.g., the number of engine failures or the number of major gifts). You can calculate the fitting ratio as:

  $$FR=N_{MC}/N_P$$

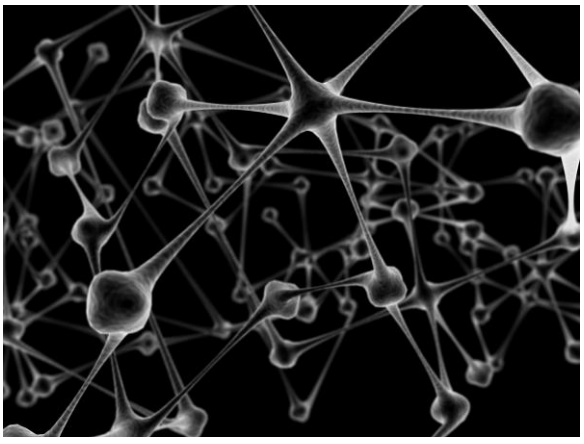
  where N_{MC} is the number of minority class records, N_P is the number of model parameters, and FR is the resulting fitting ratio. Generally speaking, to produce

really robust models – models that work well on new data, you want to work with fitting ratios greater than 100, or ideally greater than 1000.

- Third, compare the performance of your model on the training data versus the test data. If the model performs similarly on both samples (say, within +/-5%), you are probably in good shape. If you do witness a disparity greater than 5%, you are probably overfitting the data with the model. You need to make some changes. What kind of changes, you ask? The changes must **increase** the fitting ratio, which can be accomplished in two ways:
  - Decreasing the number of model parameters, N_P
  - Increasing the number of minority class records, N_{MC}

  The first option forces you to simplify your model. For instance, you might eliminate input elements in a logistic regression, or you might eliminate hidden elements in a neural network. I usually start by eliminating marginal input elements, then rebuild and retest the model. Keep in mind that, by simplifying your model, performance on the training data will go **down** – compared to your initial run. Why? Your initial run was a mirage since you overfit the data. Now, because the model has fewer parameters (and less "flexibility") to fit/overfit the training data, it is forced to generalize more. This is the goal. You'll see the positive effect as **improved** performance on the **test** data – which is what really matters.

  The latter option, increasing the number of rare event records, may require much more time and effort – but the payoff could be great. To increase the number of rare event records, you might:
  - Go to the IT staff and ask for more major gift-givers from a previous campaign that had the same appeal.
  - Look for, or create, an alternative outcome that occurs more frequently. For example, engine failures may be very rare in a regular maintenance program. But instances of degraded oil viscosity may be much more common and can act as a reasonable surrogate outcome that is a precursor to failure.



Question 3: Are new data mining/predictive analytics/modeling algorithms needed to produce better results?

No.

Oh, you'd like me to support my opinion?!? Okay. When I was in grad school, I learned a lot about various predictive modeling algorithms –programming most of them from

scratch to further my understanding. Upon leaving grad school, I went to work for an early neural network software company, and thought, "This is the greatest modeling technique there is!"

For our company, I consulted with a wide variety of clients who were using the software for all sorts of applications. Often, the client's in-house staff had been working on these applications for some time – using modeling techniques less complex than neural networks. What we found time and again was that neural networks, at best, provided a small incremental improvement in performance over the less complex methods when using the same data elements.

What did produce substantial improvements in performance?

- Improved data representation
- Additional, non-redundant data elements

The former includes transforming elements by means of ratios, products, and logarithms. The latter includes, for example, adding customer service records to transaction data for insurance policy holders.

To sum up: more data elements, from more problem dimensions, better-represented, will trump the latest, greatest modeling technique. Every time. At least, that's been my experience. See these references for others' experience[7].

That's All the Space We Have…

In this article, we looked at several challenging, frequently-asked questions:

- How much data do I need for data mining?
- How do I build robust models that perform well on test data and in the "real world"?
- Are new data mining/predictive analytics/modeling algorithms needed to produce better results?

I hope you found my responses helpful to you. In future installments, we will look at more questions from practitioners and business users alike. If you have a particular concern or question of your own, please feel free to get in touch – my contact information is below.

Tim Graettinger, Ph.D., is the President of Discovery Corps, Inc. (http://www.discoverycorpsinc.com), a Pittsburgh-area company specializing in data mining, visualization, and predictive analytics.

Your comments and questions about this article are welcome. Please contact Tim at (724)-743-3642 or tgraettinger@discoverycorpsinc.com

---

---

[1] The webinar is Data Mining: Failure to Launch – and it's free. Sorry for the shameless self-promotion.

[2] See "Grab Bag: Frequently Asked Data Mining Q & A" at http://www.discoverycorpsinc.com/grab-bag-frequently-asked-data/

[3] The concept is the same whether the data is already sitting in a data warehouse, or whether you have to collect it from scratch.

[4] If it's not clear yet, read on. And if it's still not clear after that, send me an email.

[5] Remember to count the constant term in the logistic model as an input element.

[6] The number of coefficients in a backpropagation neural network is $(N+1)\times H + H + 1$, where N is the number of input elements and H is the number of hidden-layer nodes.

[7] Here are a couple of high-profile instances where more, different data beat fancy modeling: First, related to the Netflix Prize, see http://www.hackingnetflix.com/2008/04/stanford-data-m.html. Second, related to predicting Google's earnings, see http://anand.typepad.com/datawocky/2008/04/more-data-beats.html. By the way, I'm not against sophisticated modeling at all, but my experience tells me to work with the data much more than the modeling algorithms.