Harvard Business Review

REPRINT H026NG
PUBLISHED ON HBR.ORG
JULY 07, 2015

ARTICLE
DATA

What Every Manager Should Know About Machine Learning

by Mike Yeomans

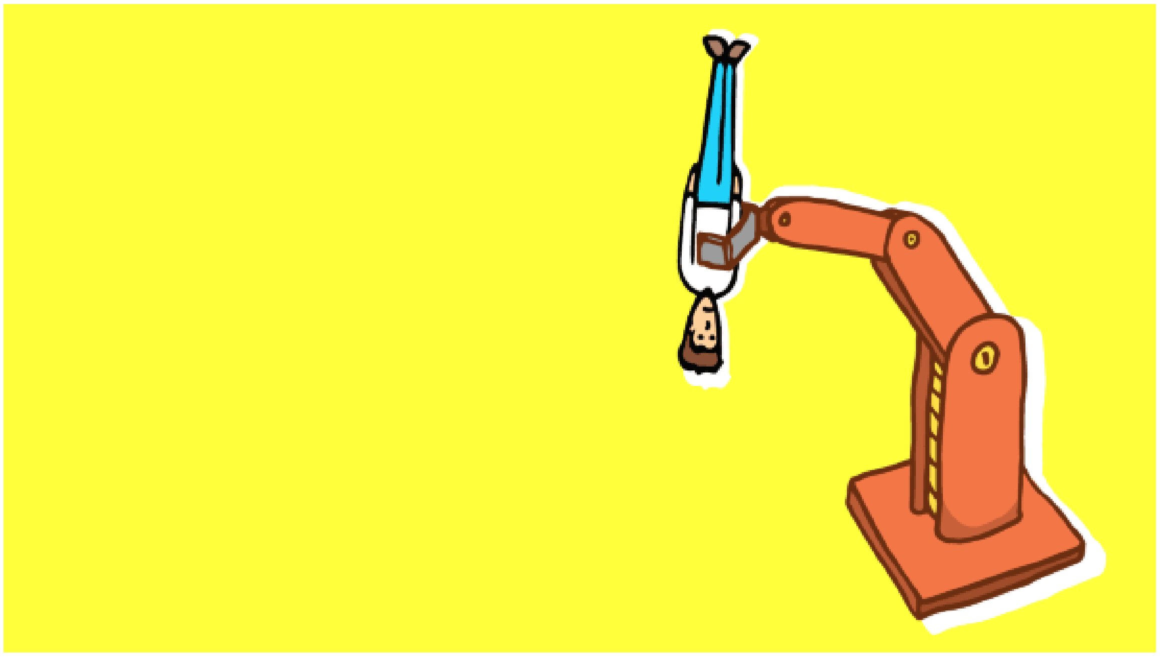
Harvard Business Review

DATA

What Every Manager Should Know About Machine Learning

by Mike Yeomans

JULY 07, 2015



Perhaps you heard recently about a new algorithm that can drive a car? Or invent a recipe? Or scan a picture and find your face in a crowd? It seems as though every week companies are finding new uses for algorithms that adapt as they encounter new data. Last year Wired quoted an ex-Google employee as saying that "Everything in the company is really driven by machine learning."

Machine learning has tremendous potential to transform companies, but in practice it's mostly far more mundane than robot drivers and chefs. Think of it simply as a branch of statistics, designed for a world of big data. Executives who want to get the most out of their companies' data should understand what it is, what it can do, and what to watch out for when using it.

Not just big data, but wide data

The enormous scale of data available to firms can pose several challenges. Of course, big data may require advanced software and hardware to handle and store it. But machine learning is about how the analysis of the data also has to adapt to the size of the dataset. This is because big data is not just *long*, but *wide* as well. For example, consider an online retailer's database of customers in a spreadsheet. Each customer gets a row, and if there are lots of customers then the dataset will be *long*. However, every variable in the data gets its own column, too, and we can now collect so much data on every customer – purchase history, browser history, mouseclicks, text from reviews – that the data are usually *wide* as well, to the point where there are even more columns than rows. Most of the tools in machine learning are designed to make better use of wide data.

Predictions, not causality

The most common application of machine learning tools is to make predictions. Here are a few examples of prediction problems in a business:

- Making personalized recommendations for customers
- Forecasting long-term customer loyalty
- Anticipating the future performance of employees
- Rating the credit risk of loan applicants

These settings share some common features. For one, they are all complex environments, where the right decision might depend on a lot of variables (which means they require "wide" data). They also have some outcome to validate the results of a prediction – like whether someone clicks on a recommended item, or whether a customer buys again. Finally, there is an important business decision to be made that requires an accurate prediction.

One important difference from traditional statistics is that you're not focused on *causality* in machine learning. That is, you might not need to know what happens when you change the environment. Instead you are focusing on *prediction,* which means you might only need a model of the environment to make the right decision. This is just like deciding whether to leave the house with an umbrella: we have to predict the weather before we decide whether to bring one. The weather forecast is very helpful but it is limited; the forecast might not tell you how clouds work, or how the umbrella works, and it won't tell you how to change the weather. The same goes for machine learning: personalized recommendations are forecasts of people's preferences, and they are helpful, even if they won't tell you why people like the things they do, or how to change what they like. If you keep these limitations in mind, the value of machine learning will be a lot more obvious.

Separating the signal from the noise

So far we've talked about when machine learning can be useful. But how is it used, in practice? It would be impossible to cover it all in one article, but roughly speaking there are three broad concepts that capture most of what goes on under the hood of a machine learning algorithm: *feature extraction*, which determines what data to use in the model; *regularization*, which determines how the data are weighted within the model; and *cross-validation*, which tests the accuracy of the model. Each of these factors helps us identify and separate "signal" (valuable, consistent relationships that we want to learn) from "noise" (random correlations that won't occur again in the future, that we want to avoid). Every dataset has a mix of signal and noise, and these concepts will help you sort through that mix to make better predictions.

Feature extraction

Think of "feature extraction" as the process of figuring out what variables the model will use. Sometimes this can simply mean dumping all the raw data straight in, but many machine learning techniques can build new variables — called "features" — which can aggregate important signals that are spread out over many variables in the raw data. In this case the signal would be too diluted to have an effect without feature extraction. One example of feature extraction is in face recognition, where the "features" are actual facial features — nose length, eye color, skin tone, etc. — that are calculated with information from many different pixels in an image. In a music store, you could have features for different genres. For instance, you could combine all the rock sales into a single feature, all the classical sales into another feature, and so on.

There are many different ways to extract features, and the most useful ones are often automated. That means that rather than hand-picking the genre for each album, you can find "clusters" of albums that tend to be bought by all the same people, and learn the "genres" from the data (and you might even discover new genres you didn't know existed). This is also very common with text data, where you can extract underlying "topics" of discussion based on which words and phrases tend to appear together in the same documents. However, domain experts can still be helpful in suggesting features, and in making sense of the clusters that the machine finds.

(Clustering is a complex problem, and sometimes these tools are used just to organize data, rather than make a prediction. This type of machine learning is called "unsupervised learning", because there is no measured outcome that is being used as a target for prediction.)

Regularization

How do you know if the features you've extracted actually reflect signal rather than noise? Intuitively, you want to tell your model to play it safe, not to jump to any conclusions. This idea is called "regularization." (The same idea is reflected in terms like "pruning", or "shrinkage", or "variable selection.") To illustrate, imagine the most conservative model possible: it would make the same prediction for everyone. In a music store, for example, this means recommending the most popular album to every person, no matter what else they liked. This approach deliberately ignores both signal and noise. At the other end of the spectrum, we could build a complex, flexible model that tries to accommodate every little quirk in a customer's data. This model would learn from both signal and noise. The problem is, if there's too much noise in your data, the flexible model could be even worse than the conservative baseline. This is called "over-fitting": the model is learning patterns that won't hold up in future cases.

Regularization is a way to split the difference between a flexible model and a conservative model, and this is usually calculated by adding a "penalty for complexity" which forces the model to stay simple. There are two kinds of effects that this penalty can have on a model. One effect, "selection", is when the algorithm focuses on only a few features that contain the best signal, and discards the others. Another effect, "shrinkage", is when the algorithm reduces each feature's influence, so that the predictions aren't overly reliant on any one feature in case it turns out to be noisy. There are many flavors of regularization, but the most popular one, called "LASSO", is a simple way to combine both selection and shrinkage, and it's probably a good default for most applications.

Cross-validation

Once you have built a model, how can you be sure it is making good predictions? The most important test is whether the model is accurate "out of sample", which is when the model is making predictions for data it has never seen before. This is important because eventually you will want to use the model to make new decisions, and you need to know it can do that reliably. However, it can be costly to run tests in the field, and you can be a lot more efficient by using the data you already have to simulate an "out of sample" test of prediction accuracy. This is most commonly done in machine learning with a process called "cross-validation".

Imagine we are building a prediction model using data on 10,000 past customers and we want to know how accurate the predictions will be for future customers. A simple way to estimate that accuracy is to randomly split the sample into two parts: a "training set" of 9,000 to build the model and a "test set" of 1,000, that is initially put aside. Once we've finished building a model with the training set, we can see how well the model predicts the outcomes in the test set, as a dry run. The most important thing is that model never sees the test set outcomes until after the model is built. This ensures that the test set is truly "held-out" data. If you don't keep a clear partition between these two, you will overestimate how good your model actually is, and this can be a very costly mistake to make.

Mistakes to avoid when using machine learning

One of the easiest traps in machine learning is to confuse a prediction model with a causal model. Humans are hard-wired to think about how to change the environment to cause an effect. In prediction problems, however, causality isn't a priority: instead we're trying to optimize a decision that depends on a stable environment. In fact, the more stable an environment, the more useful a prediction model will be.

It's important to draw a distinction between "out-of-sample" and "out-of-context". Measuring out-of-sample accuracy means that if we collect new data from the exact same environment, the model will be able to predict the outcomes well. However, there is no guarantee the model will be as useful if we move to a new environment. For example, an online store might use a database of online purchases to build a helpful model for new customers. But the exact same model may not be helpful for customers in a brick-and-mortar store – even if the product line is identical.

It's tempting to think that the sheer size of data available can get around the issue. That's not the case. Remember, these algorithms draw their power from being able to compare new cases to a large database of similar cases from the past. When you try to apply a model in a different context, the cases in the database may not be so similar any more, and what was a strength in the original context is now a liability. There's no easy answer to this problem. An out-of-context model can still be an improvement over no model at all, as long as its limitations are taken into consideration.

Even though some parts of model-building can seem automatic, it still takes a healthy dose of human judgment to figure out where a model will be useful. Furthermore, there's a lot of critical thinking that goes into making sure the built-in safeguards of regularization and cross-validation are being used the right way.

But it's also good to keep in mind that the alternative — purely human judgment — comes with its own set of biases and errors. With the right mix of technical skill and human judgment, machine learning can be a useful new tool for decision-makers trying to make sense of the inherent problems of wide data. Hopefully without creating new problems along the way.

Mike Yeomans is a post-doctoral fellow in the Department of Economics at Harvard University.