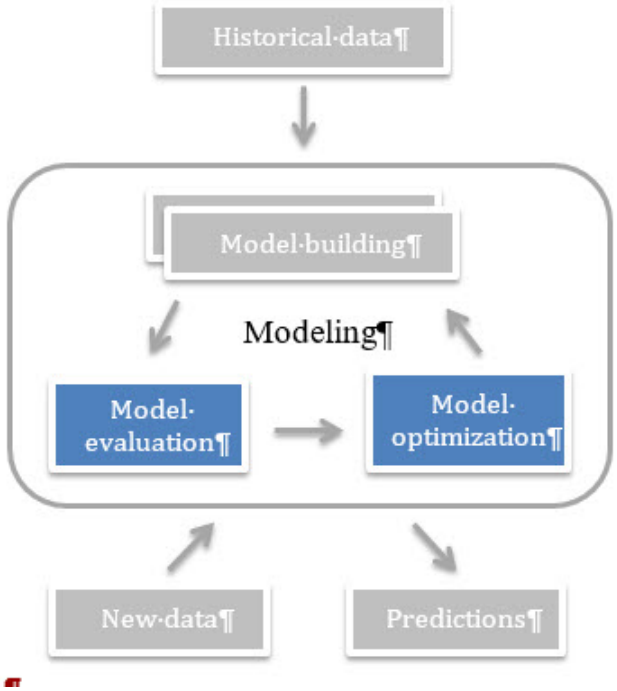
Real-World Machine Learning: Model Evaluation and Optimization

January 4, 2016
By Developer.com Staff
Send Email »

By **Henrik Brink**, **Joseph W. Richards**, and **Mark Fetherolf**



The primary goal of supervised machine learning is accurate prediction. We want our ML model to be as accurate as possible when predicting on new data (for which the target variable is unknown). Said in a different way, we want our models, which have been built from some training data, to generalize well to new data. That way, when we deploy the model in production, we can be assured that the predictions generated are of high quality.

Therefore, when we evaluate the performance of a model, we want to determine *how well that model will perform on new data*. This seemingly simple task is wrought with complications and pitfalls that can befuddle even the most experienced ML users. In this article, we describe the difficulties that arise when evaluating ML models and propose a simple workflow to overcome those menacing issues and achieve unbiased estimates of model performance.

The Problem: Over-fitting and Model Optimism

Imagine that we want to predict the production of bushels of corn per acre on a farm as a function of the proportion of that farm's planting area that was treated with a new pesticide. We have training data for 100 farms for this regression problem. Plotting the target (bushels of corn per acre) versus the feature (% of the farm treated) it is clear that an increasing, non-linear relationship exists, and that the data also have random fluctuations (see Figure 1).

To describe the challenges associated with estimating the predictive accuracy of a model, it is easiest to start with an example.

Post a comment
Email Article
Print Article

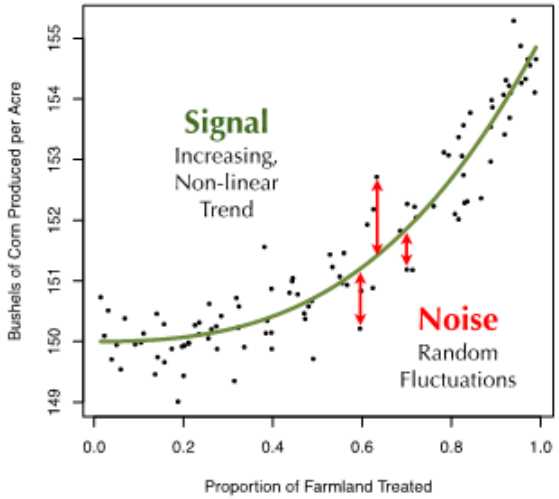


Figure 1: Signal-to-noise ratio

Now, suppose that we want to use a simple non-parametric ML regression modeling technique to build a predictive model for corn production as a function of proportion of land treated. One of the simplest ML regression models is kernel smoothing. Kernel smoothing operates by taking local averages: for each new data point, the value of the target variable is modeled as the average of the target variable for only the training data whose feature value is close to the feature value of the new data point. A single parameter, called the bandwidth parameter, controls the size of the window for the local averaging.

Related Articles

- How Hadoop is Different from Conventional BI
- Understanding BI Components and Data
- Big Data: As a Developer, Should You Care?

Figure 2 demonstrates what happens for different values of the kernel smoothing bandwidth parameters. For large values of the bandwidth, almost all of the training data are averaged together to predict the target, at each value of the input parameter. This causes the model to be very flat and to **under-fit** the obvious trend in the training data. Likewise, for very small values of the bandwidth, only one or two training instances are used to determine the model output at each feature value. Therefore, the model effectively traces every bump and wiggle in the data. This susceptibility to model the intrinsic noise in the data instead of the true signal is called **over-fitting**. Where we want to be is somewhere in the Goldilocks zone: not too under-fit and not too over-fit.

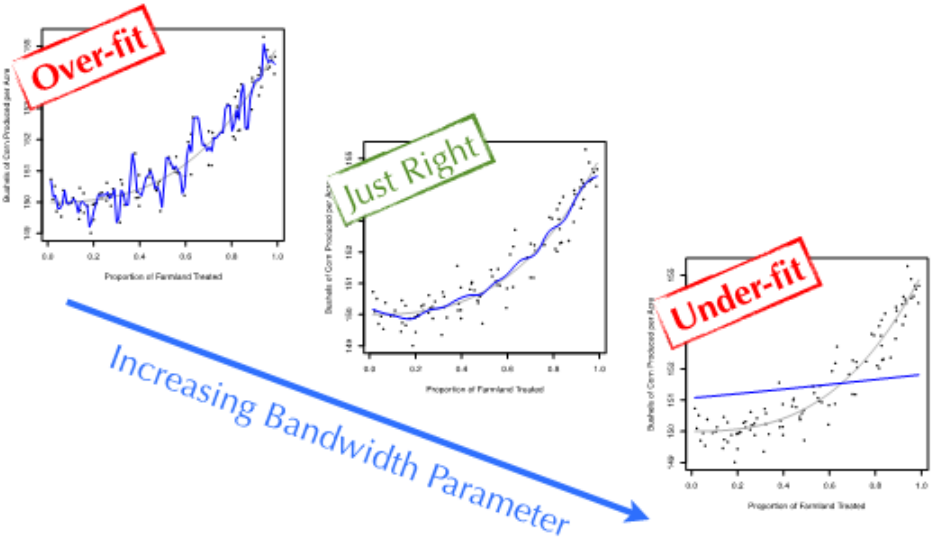


Figure 2: Three fits of a kernel smoothing regression model to the corn production training set.

Now, let's get back to the problem at hand: determining how well our ML model will generalize to predict the corn output from data on different farms. The first step in this process is to select an **evaluation metric** that captures how good our predictions are. For regression, the standard metric for evaluation is **mean squared error** (MSE), which is the average squared difference between the true value of the target variable and the model-predicted value.

This is where things get tricky. Evaluated on the training set, the error (measured by MSE) of our model predictions gets ever smaller as the bandwidth parameter decreases. This should not be unexpected: the more flexibility that we allow the model, the better it will do at tracing the patterns (both the signal and the noise) in the training data. However, the models with smallest bandwidth are severely over-fit to the training data because they trace every random fluctuation in the training set. Using these models to predict on new data will result in poor predictive accuracy because the new data will have their own unique random noise signatures that are different from those in the training set.

Thus, there is a divergence between the training set error and the **generalization error** of a ML model. This divergence is exemplified in Figure 3 on the corn production data. For small values of the bandwidth parameter, the MSE evaluated on training set is extremely small while the MSE evaluated on new data (in this case, 10,000 new instances) is much larger. Simply put, the performance of the predictions of a model evaluated on the training set is not indicative of the performance of that model on new data. Therefore, it is extremely dangerous to evaluate the performance of a model on the same data that was used to train the model.

CAUTION about Double-dipping of the training data:

Using the training data for both model fitting and evaluation purposes, can lead you to be overly optimistic of the performance of the model. This can cause you to ultimately choose a sub-optimal model which performs poorly when predicting on new data.

As we see on the corn production data, choosing the model with smallest training set MSE causes the selection of the model with smallest bandwidth. On the training set, this model yields a MSE of 0.08. However, when applied to new data, the same model yields a MSE of 0.50, which is much worse than the optimal model (bandwidth=0.12 and MSE=0.27).

We need an evaluation metric that better approximates the performance of the model on new data. This way, we can be confident about the accuracy of our model when deployed to make predictions on new data. This is the topic of the next subsection.

Figure 3: Comparison of the training set error to the error on new data for the corn production regression problem. The training set error is an overly optimistic measure of the performance of the model for new data, particularly for small values of the bandwidth parameter. Obviously, using the training set error as a surrogate for the prediction error on new data will get us into a lot of trouble.

The Solution: Cross-validation

We have diagnosed the challenge in model evaluation: the training set error is not indicative of the error of the model when applied to new data. To get a good estimate of what our error rate will be for new data, we must use more sophisticated methodology called **cross-validation**, that rigorously employs the training set to evaluate what the accuracy will be on new data.

The two most commonly used methods for cross-validation are **the holdout method** and **K-fold cross-validation**.