Testing

Alex Sverdlov
alex@theparticle.com

1 Introduction

We wish to build models that accurately describe something. We often start with data (past observations), build the model, and *hope* the model works well in predicting *future* observations.

Obviously, we don't have these *future* observations during model development—we cannot confirm how well the model works until it hits reality.

2 Training & Testing

One solution is to test the model on past observations, and see how well it does. Here we run into a few issues.

If we test the model on the same dataset that was used to train it, then we wouldn't be able to tell if the model simply memorizes the data.

One solution is to split available data into two sets: the training set, and the testing set. This is often done randomly, but may be guided with business knowledge or data availability. We can use some rule-of-thumb split, such as 80% of data for training, and 20% of data for testing (other splits as just as valid).

We train the model on the training dataset. We get training error. We test the model on the testing dataset, and we get test error—that is the number that is often reported to illustrate model accuracy (nobody reports training error).

3 Cross Validation

The static split into training set and testing set may (accidentally) put some easy instances into training-set and some hard instances into testing-set, which may skew the accuracy (in either direction).

One solution is to cut the dataset into N chunks, train on the $N-1$ chunks, and test on the remainder. We would do this N times and report the average accuracy.

This ensures that the model is trained *and* tested on each data instance—giving a more accurate reading on how the model might work on *future* observations.

The most extreme cross-validation is to train on all-but-one instance, and test on only that 1 record, and doing that for every record.

4 Holdout

One of the reasons to split available data into training-set and testing-set is to avoid biasing the model to available data (at the expense of performance on un-available future data).

As we iterate over the model during development—such as tweak model parameters, structure, etc. (hyperparameters, neural-network depth, etc.), we train on the training-set, and then test on the testing-set to see how well our tweaked parameters worked—we end up with a model that is biased towards our testing-set: our testing set determines which parameters we keep or adjust.

To avoid such bias, we often holdout a chunk of data for final testing. This chunk is not used for training, testing, or cross validation, etc., only for the final accuracy. It is important that this holdout set does not drive model development, as that will bias the model towards available data.

5 Conclusion

It is impossible to predict how accurate the model will be on previously unseen data. We *cheat* and *predict* past data to guess at what the model accuracy might be—but reality may be different.

Even if the model gets really good at predicting reality, reality may soon move away from what the model is predicting.

To maintain accuracy, model development cannot end—it should continuously incorporate newer data.

Be suspicious of accuracy numbers that are too good. Occasionally those are trainng-set numbers, or cross-validation of an overfit model without a holdout set.

Occasionally it might be more subtle, such as testing-set representing interpolation of the data—while model is tasked with extrapolation.

Also, accuracy may not be that important overall: simplicity and cost are also very important. Suppose we have a model that is not accurate, but avoids expensive mistakes: this has value, even if not accurate.