ALPINE ACADEMY

Breaking Down Prediction Errors in Machine Learning

Sung Chung, Machine Learning Engineer

Part 1: The three components of prediction errors

The most important goal of machine learning solutions is often maximizing prediction accuracy. E.g. an ideal image recognizer would correctly label previously unseen pictures. A recommender for Netflix should predict a set of unseen movies that a subscriber would enjoy watching. A stock price predictor should predict future stock prices closely more often than not.

Therefore, evaluating prediction accuracy of machine learned models is a critical part of the experimental process. The prediction error or'generalization error' evaluation often guides the model selection process as modelers explore the hyper-parameter choices and/or different model types.

Although most of machine learning practitioners are probably familiar with various techniques for estimating the prediction errors (simple division of data into training/validation sets, cross-validations, etc.), what is not as widely understood is that the evaluation process itself is an error-prone process and that there are multiple contributing 'components' to prediction errors.

Prediction errors of machine learned models have three different components -- noise, bias and variance. Here, we'll give a brief and relatively math-free description for each one.

Irreducible Noise

The noise here is defined as the noise of the label for data points that have the same features. As an example, say that you want to predict the stock price changes of companies and you are given a small number of features.

stock_delta	industry	revenue_delta	profit_delta
10	Tech	10	2
13	Tech	10	2
-12	Textile	2	-1

Given this dataset, predicting stock_delta for the first two rows will always result in errors no matter how accurate the model is since the features can't distinguish between these two data points.

I.e., noise of the error represents the 'irreducible' component of the generalization error -- no matter how hard you try, your model can't perform better (with lower error) than this. This happens for problems where fundamental noise exist in the true distributions of labels/features.

Therefore, estimating what the lower bound of a model's prediction accuracy can help data scientists in exploration. It could be largely a waste of time to try to improve a model when it is at or near the lower bound of the generalization error.

One important thing to note here, however, is that the noise for a given set of features may not really be noise in the true distribution. It is possible that given more features, for instance, the first two rows could be distinguished. In such cases, it 'may' be that we simply don't have enough features in the dataset to model the true distributions.

Bias

Most of prediction models are functions that take in inputs (features) and then spit out an output (prediction). However, it's important to know that these functions restrict how the output prediction is computed from the input features.

For instance, take a linear regression model and assume that all the input features are numbers. A linear regression model can only predict output values that can be computed as:

$$f(x) = b1 * x1 + b2 * x2 + \ldots + b0$$

where x1, x2, ... are the input feature values and b0, b1, b2, ... are the 'parameters' of the model. It is clear from the above equation then that the possible predictions for the linear regression model is limited to weighted sum of input features.

Such a model may still be reasonably accurate if the output can be approximated as a linear sum. However, if there are significantly non-linear relationships between the features and the output (for instance, say, the true relationship involves conditionals, such as 'if x1 is less than 1 and x2 is greater than 2, then output is 11, ...), this will always result in errors no matter how well we can train models or even if we have an infinite number of data points.

Such errors are called 'bias' of the model. Intuitively, these errors are the results of the machine learned model not being able to represent the true relationship between the features and the outputs, because of limits on 'forms' of calculations the model can perform.

Estimation Bias

A more detailed look into bias reveals that it's often the model 'estimation' bias that influences prediction errors in the real world. 'Estimation' bias (in some cases referred to as induction bias) refers to the fact that models generated by machine learning algorithms in practice only cover a subset of all possible configurations of the model type. I.e., the algorithm itself restricts the model configurations that can be learned.

For instance, take the ridge regression algorithm. Technically, this is also a linear regression 'learner' that has a restriction called 'L2 regularizer'. I.e., the output 'form' is still the same as ordinary linear regression models. However, the training is performed to satisfy the following 'optimization':

$$\min_a \sum_{i=1}^{n} |Y_i - a^T X_i|^2 + \lambda \|a\|^2$$

Although this may look intimidating, all it's saying is that in addition to minimizing the MSE (Mean Squared Error) of the label, it also wants to reduce the vector 'magnitude' of the coefficients (weights). The tradeoff between MSE minimization and coefficient 'magnitude' minimization is controlled through an additional user defined parameter .

Although the model's 'form' is the same as the models generated by the ordinary linear regression algorithm, the above algorithm in fact confines the possible coefficients that can be learned to much smaller ranges (whereas ordinary linear regression doesn't). Therefore, this can further increase the bias component of the prediction error.

This additional bias introduced by the 'algorithm', compared to the bias of the 'representation' is called estimation bias or induction bias. A lot of machine learning algorithms introduce estimation biases in practice.

Variance

This is perhaps the most talked about and often the dominant component of prediction errors. Variance error is what people refer to as 'overfitting' (whereas bias is known as 'underfitting'), and it refers to the error introduced by models that learn 'too much' from the given dataset.

The notion of variance in prediction error is closely related to variance of samples in statistics. A dataset one has for training models is often just a 'sample' of the underlying 'true' data. As an example, say that the world has exactly 50% males and 50% females. If we choose 1000 people randomly from the world population, then it's highly unlikely that we will get exactly 500 males and 500 females.

It's possible, for instance, we might end up with 600 males and 400 females, among many other possibilities. If our goal was to 'learn' the ratio between males and females in the world from the 1000 sample points, then there's a very good chance we will end up with an error.

The exact same thing happens when building prediction models. The training dataset is just a sample of the true data, as mentioned earlier. The error due

to variance happens when the training dataset label/feature distributions are different from the true distributions and when the model trainer learns too 'eagerly' to predict correctly on the somewhat incorrect training dataset.

Part 2: Methods for estimating prediction errors

Estimating Prediction Errors

In the previous post, we described how three different error components may contribute to the overall prediction errors. Here, we will cover different methods of estimating prediction errors and its noise, bias, and variance. Estimating the prediction error and its noise, bias and variance is a statistic estimation problem where simple approaches may yield incorrect estimates.

For instance, one of the most popular approaches to estimating prediction errors is to divide up the given dataset into a training dataset and a validation dataset. However, if we only do this once, then luck may play a significant role in our estimation.

First, in such scenarios, we are essentially training a model on a single sample and validating the model on a single sample. Then there's a chance that, if the prediction estimation comes out to be very good, it was luck and that against new dataset, it'll perform poorly. For example, it may be that the validation sample and the training sample are both very different from the true distributions and yet similar to each other.

There are several approaches that one can use to mitigate such 'luck' playing a significant role in this. A relatively easy way, particularly with the abundance of data nowadays, is simply to use a lot more training data points and a lot more validation data points. The law of large numbers dictates that our estimates will be more accurate and less 'lucky' with larger sample sizes.

If drawing larger samples is not plausible, there are various resampling strategies (cross-validations, bootstrapping, etc.) that yield much more accurate estimates of prediction errors. Additionally, resampling strategy is a viable way of estimating bias and variance of prediction errors.

Estimating Bias, Variance, and Noise

When we mention that prediction errors have bias, variance and noise components, it's important to note that we are talking about 'on-the-average' behavior of the models that have been trained with different samples of the same sizes.

E.g., if we train just one model on a training sample, it may turn out that the training sample is a very close approximation of the true data and the model we trained has very low overfitting error. However, for the other 99 training samples, the overfitting errors may be high. Variance in ML prediction is this variance of errors across multiple training samples.

Finding this 'average' error prediction is important since we usually don't know whether we are lucky with our training sample draw or not.

One way we can estimate bias and variance is through resampling techniques. E.g. we can bootstrap training samples repeatedly and use OOB (out of bag) samples to estimate their average predictions and subsequently compute bias and variance of the predictions (for mathematical definitions in case of linear regression, go here (page 223, equation 7.9)).

Such resampling techniques are expensive computationally since they involve training models many times on the same dataset. Moreover, often one wants to measure bias/variance breakdowns across different models or different hyperparameters. However, with the availability of large clusters of machines nowadays, it's not an impractical proposition any more.

Error Measures and Bias, Variance, and Noise

Figuring out how bias, variance and noise add up to the overall prediction error is a challenge in itself. It turns out that this also depends on how the prediction errors are 'measured'.

For example, if the errors are measured in MSE (Mean Squared Error), then the average prediction error is additive. I.e. :

$$PredErr = irreducible\ noise + bias2 + variance$$

(Exact mathematical definition is here (page 223, equation 7.9))

What this means is that, in case of regression, we can guarantee reduction of the overall prediction error through bias reduction, variance reduction, or both.

However, if we measure the error through mis-classification rate, as in many classification problems, the effect of individual component on the overall prediction error is in fact non-linear. For example, classification errors can be shown to be:

$$MisclassificationErr = c1 * irreducible\ noise + bias + c2 * variance$$

where c1 depends on the proportion of correctly classified data points and incorrectly classified data points for the same features, and c2 depends on bias. (Exact mathematical definition can be found in this paper (page 24))

Intuitively, this is because with classification errors, 'wrongly right prediction' is possible. E.g., if there's noise in the labels, then incorrectly predicting on a data point whose label is different from the 'consensus label' can mitigate the overall noise contribution. For example, say, we have a set of feature values and 60% of times, its label comes out to be one. The remaining 40% of times, the label for those feature values comes out to be zero. Then, we can define the misclassification noise to be 2 / 5 for this set of feature values.

Now, if we train a model on a random training sample and its prediction is always 1, then the contribution from the noise to the average prediction error would be unchanged at 2 / 5. However, if the prediction sometimes comes out as 1 on some training data, and sometimes comes out as 0 on different training data, then the noise contributes to the error only when the prediction is 1.

Likewise, if there's error due to the bias of the model (the average prediction for a feature set is different from the consensus label), then the error due to overfitting may in fact reduce the overall error since a different prediction from the average prediction may in fact be the correct classification.

The overall classification error rate can still be reduced by reducing bias and/or variance. However, their relationships to the overall error rate are not linear as in the MSE models.

Part 3: Model Selection

In the previous post, we covered methods for estimating overall prediction errors and their component parts. In this last post, we will discuss model selection.

Model Selection and Data Size

In model selection, estimating bias and variance is especially useful. Knowing whether the prediction errors we are getting are due to bias or variance can lead us to explore proper models (e.g., if the errors are mostly bias, then we can explore high variance, low bias models. If the errors are mostly variance, then we can explore low variance, high bias models.)

In a 'big data' world, high variance, low bias models are often preferred. This is because the larger the dataset size is, the smaller the variance of error will be. This is because there's a better chance that the dataset would more closely resemble the underlying distribution the larger the dataset is. As the size of the dataset goes to infinity, the variance would entirely disappear.

In contrast, the bias is irreducible even if one has a lot of data points. Intuitively, this makes sense since bias is due to the fundamental restrictions on the 'form' of the output calculations. If the true underlying relationship between features and the prediction can't be represented by the model, it'll always have the bias error, even with an infinite amount of data points.

The notion of 'big' is, however, also dependent on the problem domain. E.g., 10 million data points may be relatively small in image/speech/NLP domains, whereas they may be plentiful in a preference prediction problem with only a dozen features or so.

Examples of high variance models may be:

- linear model with a large number of features that include many expanded features (such as polynomial expansion).
- Decision-trees without bounds on their sizes.
- Deep neural networks.

Model Selection through Bias-Variance Tradeoffs

A lot of times, there's a trade-off relationship between bias and variance components of prediction errors. E.g., when one reduces the bias of a prediction error, there's a good chance that the variance of the error would go up. When one reduces the variance, the bias tends to go up. The job of a data scientist often comes down to finding the 'optimal' balance between bias and variance to minimize the overall error rate.

Intuitively, this is because bias and variance are both related to flexibility (or complexity) of models. The more flexible (or complex) a model is, the smaller its bias would be since the model might be able to induce any relationship between label and features from a training sample. But at the same time, such flexible models would typically 'eager-fit' to any training sample, giving a high variance error. Vice versa is also true for inflexible models.

The act of hyper-parameter search and feature selection (such as finding optimal for regularized models, finding the optimal tree depths, finding the proper number of features, etc.) is precisely the act of finding the optimal balance between bias and variance.

Ensemble models that achieve the best of both worlds

There are ensemble techniques that are capable of 'getting the best of both worlds'. The most famous one is Random Forest. It is based on 'averaging' predictions of multiple decorrelated trees that are individually highly overfit. It turns out that because individual trees are highly overfit, the overall model's bias is very low to begin with. And by adding up and averaging many trees'

predictions that are different from each other, the variance can be reduced without increasing bias! In some counterintuitive fashion, this is a technique where adding more complexities to a model can actually decrease both bias and variance.

There is also a recently developed variant of deep neural network called drop-out network that leverages a similar idea and has been shown to perform extremely well.

As a side note, one thing people often seem to get confused about is the difference between Random Forest and boosting. Whereas Random Forest can reduce both bias and variance as described above, boosting is an ensemble technique where one starts with a weak, very high-bias and low-variance model and as more 'weak' models are added, the overall ensemble moves to a lower-bias and higher-variance state. Therefore, one still has to search for the optimal balance between bias and variance with boosting, whereas with Random Forest, one can often indefinitely add more trees to reduce variance without increasing bias (the error reduction would flatten after a certain number of trees, but it'll not go back up with more trees).

This is one of the reasons that Random Forest is extremely popular, since one can get highly accurate models without a lot of 'parameter-search' as he/she has to do with other models. (However, there are still certain parameters that need to be tweaked, such as the number of random features to be searched per node, and it turns out that this can still affect 'bias/variance' of the model).