A Bias VS. Variance

Report Issue (https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

In this article, we will talk about the notions of `bias and variance`, which provides another perspective to look at the phenomenon of **underfitting** and **overfitting** that we discussed before.

Before we start, we would like to put a statement below. It probably does not ring a bell for you at the moment, but hopefully you would get the answer yourself once you go through the article.

> Bias is a learner's tendency to consistently learn the same wrong thing. Variance is the tendency to learn random things unrelated to the real signal **[1]**.

In order to define bias and variance, we need to first define the notion of `main prediction` **[2]**. For those of you who are already familiar with the concepts of model and loss function, you can skip the part of definitions, and jump directly to the section of Main Prediction. For the sake of completeness, we first give the definitions of basic concepts before diving into the main subject of this article.

Definitions

Given a training set $s = \{(\vec{x_1}, t_1), ...(\vec{x_n}, t_n)\}$, a learner (a machine learning algorithm) produces a model F. Given a test example $\vec{x_k}$, this model produces a **prediction** $y_k = F(\vec{x_k})$. Each sample in the training set consists of two elements, $\vec{x_i}$ is a vector of attributes associated with the sample, and t_i is the target attribute to predict for the sample.

For example, a train sample might look like: $\vec{x_i}$ = (type='appartment', location='LA', surface='120m^2'), t_i = (price='420,000\$'). The task of a learner is then to predict the price of an estate given its properties.

Given an training sample $(\vec{x_i}, t_i)$, the learner produces a prediction as $F(\vec{x_i})$, we then define the **loss function** $L(F(\vec{x_i}), t_i)$ as the cost that is incurred by the difference between the prediction $F(\vec{x_i})$ and the true value t_i associated with the sample. The larger the difference, the bigger the loss. If the target attribute t_i is of numerical value, a common loss function would be a **square error**, *i.e.* $L(F(\vec{x_i}), t_i) = (F(\vec{x_i}) - t_i)^2$. Following the above example, if a model F produces the price of the above example $\vec{x_i}$ as '410,000\$', then the result of the loss function would be $(420,000 - 410,000)^2 = 10^8$.

Main Prediction

> Given a loss function L and sets of training set $S = \{s_1, s_2...s_n\}$, the main prediction for a learner is defined as $y_m = argmin_{y'} E_S(L(y, y'))$.

For each training set s_i, we train a model F_i with a given learner. For a given training sample, we then produce a set of predictions $Y = \{y_1, y_2...y_n\}$ with y_i corresponding to the result produced by the model F_i. The ***main prediction*** y_m is the prediction y' whose average loss with regards to all the predictions in Y is minimum (*i.e.*, it is the prediction that "differs least" from all the predictions in Y according to L). **[2]**

In a particular case, when the loss function is square error, *i.e.* $L(y, y') = (y - y')^2$, the main prediction with regards to the entire training sets S, is then **the mean of the predictions**, *i.e.* $y_m = E_S(Y) = \frac{1}{n}\sum_{i=1}^{n} y_i$. One can refer to the paper **[2]** in the references for the deduction process.

> We can interpret the main prediction as the expect answer for a given training sample, from a given learner (a machine learning algorithm).

To better understand the concept of the main prediction, let's imagine that a learner is playing a dart-throwing game, *i.e.* a learner is a player. Each time a player throws a dart, it involves two corresponding activities: *1).* the player poses and aims, *i.e.* the learner trains a model from a given training dataset. *2).* the player throws the dart, *i.e.* we say that the model trained by the learner produces a prediction. Intuitively, the bullseye is the target of the prediction. The closer a dart to the bullseyes, the better the player (learner) is.

Then each time before the player throws a dart, we can ask ourselves a question: *how many points that the player would score*. And this is where the concept of the main prediction comes into the picture. As it turns out that the best guess that one can bet on is the main prediction of this learner (player), which could be approximated by the average score that the player has achieved during all past games. For example, we can bet that a dart out of the hand of a good player would not stray too far away from the bullseye.

> Intuitively, we can consider the main prediction as the general ***tendency*** (performance) of a learner, *i.e.* expected points that a player can score in a game.

With the notion of the main prediction, we now can define the notions of bias and variance.

Bias

> The bias of a learner (algorithm) on the example $(\vec{x_i}, t_i)$ is defined as $B(\vec{x_i}) = L(y_m, t_i)$, where y_m is the main prediction, t_i is the target value, and L is the loss function.

In a word, the bias of a learner, with regards to an example, is the loss incurred by the difference between the main prediction (the general tendency) and the actual value of the target attribute (target value).

The closer the main prediction to the target value, the less the loss. The more likely a learner produces a model that gives the right prediction, the less bias the learner has.

According to the definition, the main prediction of a learner is the mean prediction of all models that are produced by the learner from various training sets. Therefore, given a learner, with infinite training sets, one can assume that the main prediction would converge to a constant value that is linked with the nature of the learner. When the bias of a learner is zero, we can say that the learner can produce a number of models whose mean prediction is the desired target value **[2]**.

If a learner has a relatively high bias and the learner does indeed learn something from the data, then we can say that the model resulted from the learner has a high tendency to produce an erroneous prediction.

Variance

> The variance of a learner (algorithm) on the example $(\vec{x_i}, t_i)$ is defined as $V(\vec{x_i}) = E_S(L(y_m, y))$, where y_m is the main prediction, y is a prediction produced from a model that is trained on a training set $s \in S$, L is the loss function, and E_S is the average function over the list of loss values.

In a word, $V(\vec{x_i})$ measures the loss incurred by its fluctuation around the main prediction in response to different training sets.

The more stable the performance of a learner, the less its variance. In the game of dart, a learner with *high variance* corresponds to a lousy player who rarely lands the dart in the same place. On the other hand, a learner with low variance could correspond to a decent player who rarely misses the bullseye.

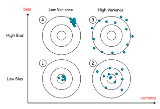
The variance is independent of the true value of the predicted variable, and is zero for a learner that always makes the same prediction regardless of the training set **[2]**.

If a learner has high variance, then we would know that the learner is highly sensitive to the training data set, *i.e.* a minor noise in the training data could lead to an ill-formed model that produces wrong predictions.

Bias-Variance Coordinate

> The bias and variance can serve as metrics to evaluate the performance of a learner.

Below we draw a plot with two dimensions: the X dimension indicates the variance of a learner, and the Y dimension indicates the bias of a learner. Following the dart game example that we discussed in previous sections, the learner assumes the role of a dart player. Each dot on the board corresponds to a prediction made by the learner. The closer is the dot to the bullseye, the closer is the prediction to the target value. We then can classify the learners into four different types as follows:



- (1). An ***ideal learner*** would situate at the low-left part of the plot, where we find the low bias as well as low variance. This is the decent dart player that rarely misses the bullseye. A good player never makes mistakes, *i.e.* a good learner always makes good predictions.

- (2). Now, moving to the right-hand side of the ideal learner, we have a ***fair learner*** who manages to score some points (*i.e.* low bias), yet the darts are all over the places (*i.e.* high variance). The learners that are situated in this area are usually complex algorithms that could manage to train some decent models sometimes, but in general, the performance of the model is not too promising. The case where we do not obtain a good model, is also called **overfitting**, *i.e.* the model pays too much attention to the irrelevant noise.

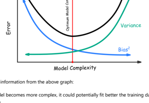
- (3). We then move to the up-right part of the plot, here we find the ***terrible learner***, who has both high bias and high variance. The terrible learner is not able to extract information from the data and basically learns nothing. The prediction that learner produces is not relevant (high bias), and what's even worse, the learner is not consistent with its strategy but making a random guess (high variance).

- (4). Right next to the terrible learner, we find the ***naive learner***, who has high bias yet low variance. The naive learner often adopts some straightforward strategies, which could explain why it produces stable outputs (low variance). But the strategy is too simple to capture the essential information from the data, which results in producing a **underfitted** model.

Bias-Variance Tradeoff

> Bias is reduced and variance is increased in relation to the complexity of the models produced by a learner.

The correlation between the model complexity and bias-variance can be described in the following graph:



One can draw a few information from the above graph:

- When the model becomes more complex, it could potentially fit better the training data set, as a result, the bias is reduced.
- Meanwhile, when the model becomes more complex, the variance increases, since the model becomes more sensitive to the noise in the data.
- As one can see, the total error of a model is correlated to the components of Bias^2 and Variance.

Indeed, when the loss function of the model is square error, its total error can be decomposed as:

$$Err(x_i) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

We skip the deduction process for the above formula, but one can refer to the page here (https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff#Derivation) for more details.

From the above definitions, we can see that a good learner should have low bias and also low variance. However, it is difficult to have them both, since these two properties are contradicting to each other. As a result, one needs to find the sweet spot on the model complexity in order to obtain the best result.

> Given a learner, often one can adjust its bias and variance, by tuning the parameters involved in the learner.

For example, if we construct a decision tree for a classification problem, without any constraint the decision tree could overgrow itself to fit every bit of the training data set, including the noisy data. As a result, we might obtain a decision tree model with a low bias for the given training data set. However, it could end up with high bias as well as high variance for the unseen data sets, since it ***overfits*** the training data set. To mitigate the problem, one can impose some constraints to limit the growth of a decision tree, *e.g.* setting the max depth a tree can grow, which might result in higher bias in the training data set. In exchange, we could obtain a model with lower variance in the unseen data set, as well as lower bias, since the model is trained to be more generalized.

Conclusion

In this article, we clarify the notions of bias and variance, which are the characteristics associated with a learner (*i.e.* a machine learning algorithm). These characteristics are exhibited in the scenarios of applying the learner to solve a particular machine learning problem. Therefore, to measure the bias and variance, one should apply the learner in a set of training datasets for a given problem.

The bias and variance of a learner are defined under the context of the problem, *i.e.* the training data sets, the learning task and the loss function *etc.* which vary from problem to problem. In general, it is not fair to say that a learner is of high bias or of high variance, without mentioning the context. For example, the linear regression algorithm might be a terrible learner (high bias and high variance) for the image classification problems, while it could excel in some simple classification problems whose data sets contain only a few attributes.

Given a problem, the bias and variance of a learner is often not fixed either. One could tune the parameters with the learner to strike a balance between the bias and variance. Overall, when the complexity of the models produced by a learner increases, the bias of the learner decreases, while its variance increases.

Finally, let's conclude the article with the statement that we put at the beginning:

> Bias is a learner's tendency to consistently learn the same wrong thing. Variance is the tendency to learn random things unrelated to the real signal [1].

Now, if you have reached this point, you should be able to make sense of the statement. If you have any doubts or questions, feel free to pose in the Discussion (https://leetcode.com/discuss/explore/machine-learning-101) forum.

Further Readings

- [1]. *A Few Useful Things to Know about Machine Learning* (https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf). Pedro Domingo. 2012. University of Washington.

- [2]. *A Unified Bias-Variance Decomposition and its Applications* (https://homes.cs.washington.edu/~pedrod/papers/mlc00a.pdf). Pedro Domingo. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238, Stanford, CA, 2000. Morgan Kaufmann.