STATISTICS: BIAS VARIANCE TRADE-OFF

1. Trade-off between Bias and Variance

As we saw in Regression Analysis, when we have several independent variables in our data set, we can fit several models. In addition, we also saw that the relationships between some of the independent variables and the response variable may or may not be statistically significant. We were fitting linear models of the form

$$f(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \tag{1}$$

where there are p independent variables. This is the familiar $Ax = b$ form and we saw how we could use Ordinary Least Squares to fit the best possible solution in the least sum of squared error-sense.

We can fit models that are not limited to linear powers of the variables. We can fit higher-order terms such as quadratic, cubic terms etc. We can also fit interaction terms such as $x_i \times x_j$, for instance. A more powerful model might have the following form:

$$f(x) = \beta_0 + \sum_i \beta_i x_i + \sum_i \sum_j \beta_{ij} x_i x_j \tag{2}$$

Here, we introduced quadratic terms. But, we are not limited to that. In general, we can fit a polynomial function of an arbitrary degree, limited only by the number of data points we have. If we have N data points, we can always fit an $N-1$ degree polynomial which will have zero error with respect to the N data points but may not have any explanatory power in that it may not capture the form of the function y appropriately.

In general, polynomial functions are more powerful than simple linear functions (of the form shown in Equation 1) and we can find a non-linear function that has zero error with respect to the N data points we have access to. Does this mean that polynomial (in general, non-linear) functions are always better than linear functions? Shouldn't we always consider a function that minimizes the error to zero? We'll answer this by considering the types of errors a model makes. This is a very important concept in statistical learning and Machine Learning.

We made an implicit assumption that the response variable y is a function of x plus noise ϵ. That is y can be modeled as

$$y = f(x) + \epsilon \tag{3}$$

where $f(x)$ is shown in Equation 1. In general, the noise term can be thought to comprise two terms: a reducible component because of the model $f(x)$ not being an accurate representation of y; and an irreducible component that represents measurement noise that is beyond our control. The best we can hope for is to eliminate the reducible error by fitting a better model. If we fit a model that reduces the error completely in the data set we have access to (also known as the training set in Machine Learning settings), we find that the model has ended up fitting to the irrelevant details introduced by the irreducible errors and therefore is needlessly complex and has limited value.

Let us assume that we have N data points of the form $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_N}, y_N)\}$ where y is a scalar response variable and $\mathbf{x}$ is a p dimensional vector. The response variable y has the following functional form: $y = f(x) + \epsilon$ where $E(\epsilon) = 0$. That is, the noise ϵ is a random variable with a zero mean. Given the data set D comprising these N points, we estimate a model $\hat{f}(x|D)$ which approximates $f(x)$. Let f_i represent $f(x_i)$ and $\hat{f}_i$ represent $\hat{f}(x_i)$ where $x_i \in D$.

We want to fit a model that minimizes the expected value of the squared error. That is, we want to fit a model so that it has the lowest expected error with respect to the true underlying function. The mean squared error is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_i)^2 \tag{4}$$

We don't want to lower the error with respect to only this particular data set D, but across all possible data that can be generated by the true underlying function $f(x)$. That's why we are interested in the minimum value of E_{MSE}, the expected value of the MSE.

$$E_{MSE} = \frac{1}{N}\sum_{i=1}^{N}E[(y_i - \hat{f}_i)^2] \tag{5}$$

We know that $y_i = f_i + \epsilon$, where f_i is the true function and ϵ is the irreducible error that cannot be eliminated and inherent in our measurements. $E[(y_i - \hat{f}_i)^2]$ can be split as $E[(y_i - f_i + f_i - \hat{f}_i)^2]$. This expands to:

$$E[(y_i - \hat{f}_i)^2] = E[(y_i - f_i)^2] + E[(f_i - \hat{f}_i)^2] + 2E[(f_i - \hat{f}_i)(y_i - f_i)] \tag{6}$$

and continuing to

$$E[\epsilon^2] + E[(f_i - \hat{f}_i)^2] + 2E[f_i y_i] + 2E[f_i \hat{f}_i] - 2E[f_i^2] - 2E[\hat{f}_i y_i] \tag{7}$$

Since f_i is a deterministic function, its Expected value is f_i. Similarly, $E(y_i)$ is f_i as $y_i = f_i + \epsilon$ and $E(\epsilon)$ is zero by definition. Similarly $E[\hat{f}_i y_i] = E[\hat{f}_i(f_i + \epsilon)] = E[\hat{f}_i(f_i]$. This cancels out the other term. So, we are left with

$$E_{MSE} = E[\epsilon^2] + E[(f_i - \hat{f}_i)^2] \tag{8}$$

Using a similar procedure, $E[(f_i - \hat{f}_i)^2]$ can be written down as $E[(f_i - E[\hat{f}_i])^2] + E[(E(\hat{f}_i) - \hat{f}_i)^2]$. Where $E[\hat{f}_i]$ is the expected value of the fitted function at data point i. We call this the bias of the fitted function.The second term is the variance of the fitted function at the data point i. So, we can decompose

$$E_{MSE} = Var(\epsilon) + (bias(\hat{f}_i))^2 + Var(\hat{f}_i) \tag{9}$$

So, there is an irreducible error and then there are two additional error terms due to the model. The first one is due the a bias in the model (or a systematic departure of the model from the underlying true function). The final term is due to the variance of the model in not predicting a particular data set well.

In general, non-linear models such as polynomial fits will have low bias because they are able to go arbitrarily close to f_i. However, they introduce a higher variance in the model as they may not fit any particular instance of the underlying data well. When we have a high variance but low bias, we call it *overfitting* and when we have a model with a high bias but low variance, we call it *underfitting*. The goal of statistical learning is to find a model that neither overfits not underfits. Typically, if we try to reduce bias, we'll increase the variance and if we try to reduce variance, we'll increase bias. This leads to the bias-variance tradeoff. So, when we try to fit models, we trade off bias against variance and try to fit the model that gives good prediction overall by balancing bias and variance.

One way to find a good balance between bias and variance is to perform a procedure known as cross-validation. In cross-validation, rather than fitting your function on the entire N, you leave a small portion out (typically a 10-20%) and you learn the function (that is, perform Ordinary Least Squares or any other type of function fitting). You then evaluate your function on the remaining data (the validation set). This will give you an estimate of the fitting error. You repeat this procedure several times, each time taking a different random slice of the data for the validation set and the remaining data as your training set. This average of several cross-validation iterations gives a better estimate of the fitting error than taking only one cross-validation iteration. This procedure is then repeated for all different models (e.g. different degrees of the polynomials). If we plot the mean cross validation error as a function of the complexity of the model (e.g. the degree of the polynomial), we'll see a characteristic U-shaped curve where the error initially goes down as the bias decreases and then increases again as the bias has gone to zero but the variance starts increasing as the model complexity starts growing. The model parameter setting at which the bias and variance are both minimized represents a good model choice trading off bias and variance.