2024/09/18

Bias and Variance

1. Theoretical Minimum: Bias and Variance[1]

Bias and variance are two major sources of errors **that prevent supervised learning algorithms from generalizing beyond their training set**. Let us denote the error within the sample (training) E_{in} and the error of new data (out-of-sample) E_{out}. We are interested in the source of errors of E_{out}.

The bias-variance decomposition of out-of-sample error is based on squared error measures. The out-of-sample error is:

(1) $E_{out}\left(g^{(\mathcal{D})}\right) = \mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$

where the $\mathbb{E}_{\mathbf{x}}$ denotes the expected value with respect to **x** (based on the probability distribution on the input space $\mathcal{X}$). We have made explicit the dependence of the final hypothesis g on the data $\mathcal{D}$, as this will play a key role in the current analysis.

We can rid Equation (1) of <u>the dependence on a particular data set by taking the expectation with respect to all data sets</u>. We then get the expected out-of-sample error for our learning model, independent of any particular realization of the data set,

(2)

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{out}\left(g^{(\mathcal{D})}\right)\right] &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})^2\right] - 2\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})\right]f(\mathbf{x}) + f(\mathbf{x})^2\right]\end{aligned}$$

The term $\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})^2\right]$ gives an 'average function', which we denote by $\bar{g}(\mathbf{x})$. One can interpret $\bar{g}(\mathbf{x})$ in the following operational way. Generate many data sets $\mathcal{D}_1, \cdots, \mathcal{D}_K$ and apply the learning algorithm to each data set to produce final hypotheses $g_1, \cdots, g_K$. We can estimate the average function for any **x** (based on the probability distribution on the input space $\mathcal{X}$) by:

[1] Some materials are taken from Abu-Mostafa, Y S, Magdon-Ismail, M., Lin, H-T (2012) *Learning from Data*, AMLbook.com.

$$\bar{g}(\mathbf{x}) \approx \frac{1}{K}\sum_{k=1}^{K} g_k(\mathbf{x})$$

We can now rewrite the expected out-of-sample error in terms of $\bar{g}(\mathbf{x})$:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{out}\left(g^{(\mathcal{D})}\right)\right] &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})^2\right] - 2\bar{g}(\mathbf{x})f(\mathbf{x}) + f(\mathbf{x})^2\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})^2\right] - \bar{g}(\mathbf{x})^2 + \bar{g}(\mathbf{x})^2 - 2\bar{g}(\mathbf{x})f(\mathbf{x}) + f(\mathbf{x})^2\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right] + \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\end{aligned}$$

The term $\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2$ measures how much the average function that we would learn using different data sets $\mathcal{D}$ deviates from the target function $f(\mathbf{x})$ that generated these data sets. This term is appropriately called the **bias**:

$$\text{bias}(\mathbf{x}) = \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2$$

as it measures how much our learning model **is biased away** from the target function $f(\mathbf{x})$. This is because $\bar{g}$ has the benefit of learning from an unlimited number of data sets, so **it is only limited in its ability to approximate f by the limitation in the learning model itself.**

The term $\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]$ is the **variance** of $g^{(\mathcal{D})}(\mathbf{x})$:

$$\text{var}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]$$

which measures the variation in the final hypothesis, depending on the data set.

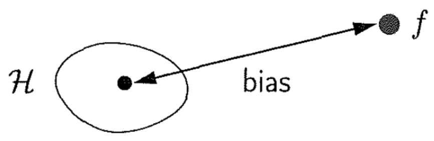
We thus arrive at the bias-variance decomposition of error:

(2) $\mathbb{E}_{\mathcal{D}}\left[E_{out}\left(g^{(\mathcal{D})}\right)\right] = \mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x}) + \text{var}(\mathbf{x})] = \text{bias} + \text{var}$

where $\text{bias} = \mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})]$ and $\text{var} = \mathbb{E}_{\mathbf{x}}[\text{var}(\mathbf{x})]$.

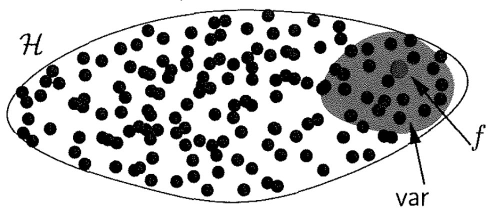
Summary: bias captures the sense that, even if all possible data were presented to the learning method, it would still differ from the target function by this amount. On the other hand, var shows the variation in the final hypothesis, depending on the training data set, notwithstanding the target function

The approximation-generalization tradeoff is captured in the bias-variance decomposition. To illustrate, let's consider two extreme cases: a very small model (with one hypothesis) and a very large one with all hypotheses.



Very small model. Since there is only one hypothesis, both the average function $\bar{g}$ and the final hypothesis $g^{(\mathcal{D})}$ will be the same, for any data set.

Thus, var = 0. The bias will depend solely on how well this single hypothesis approximates the target f, and unless we are extremely lucky, we expect a large bias.



Very large model. The target function is in $\mathcal{H}$. Different data sets will lead to different hypotheses that agree with f on the data set, and are spread around f in the gray region.

Thus, bias $\approx$ 0 because $\bar{g}$ is likely to be close to f. The var is large (heuristically represented by the size of the red region in the figure). This (low bias and high var) is the typical scenario we encountered when we apply powerful machine learning models (e.g., gradient boosted tree, neural networks) to our data.

2. Example: Bias and Variance

Let's construct an example to make the theory concrete. Suppose we have a hypothesis set consisting of all linear regressions without an intercept term, $h(x) = ax$. The input variable x is uniformly distributed in the interval $[-1, +1]$. The training data $\mathcal{D}$ consists of only <u>two points</u> $\{x_1, x_2\}$. The target function $f(x) = \sin(\pi x)$. Thus, the full data set is $\mathcal{D} = \{(x_1, \sin(\pi x_1)), (x_2, \sin(\pi x_2))\}$. The learning algorithm returns the line fitting these two points as $g^{(\mathcal{D})}$ ($\mathcal{H}$ consists of functions of the form $h(x) = ax$). We are interested in finding the bias and variance.

Q: How to compute the bias and variance?

A: To compute the bias and variance, we need to compute $g^{(\mathcal{D})}(x)$ and $\bar{g}(x)$.

$g^{(\mathcal{D})}(x)$:

$g^{(\mathcal{D})}(x)$ can be obtained by generating two points between the interval $[-1, +1]$ randomly and use `LinearRegression` from `Scikit-Learn` to find $g^{(\mathcal{D})}$ for this dataset. The following code shows how to construct the training data,

```
import numpy as np
from scipy import stats
def gen_sindata(n=2):
    x = stats.uniform(-1,2) # define random variable
    v = x.rvs((n,1)) # generate sample
    y = np.sin(np.pi*v) # use sample for sine
    return (v,y)
```

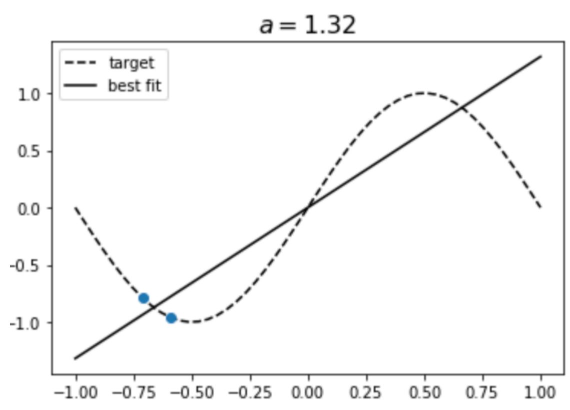
Using `Scikit-learn`'s `LinearRegression` object, we can compute the a parameter for a single hypothesis. Note that we have to set `fit_intercept=False` keyword to suppress the default automatic fitting of the intercept.

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression(fit_intercept=False)
X, y = gen_sindata(2)
lr.fit(X,y)
```

And plot $g^{(\mathcal{D})}$ for this training data (your plot might be different from mine as the two points are generated randomly):

```
%matplotlib inline
import matplotlib.pyplot as plt
xi= np.linspace(-1,1,50)
yi= np.sin(np.pi*xi)
y_pred = lr.predict(xi.reshape(-1,1))

plt.plot(xi, yi,'--k',label='target')
plt.plot(X, y, 'o')
plt.plot(xi, y_pred, c='k', label='best fit')
plt.legend(loc='best')
plt.title('$a=%3.3g$'%(lr.coef_),fontsize=16)
```



$\bar{g}(x)$:

In this case, $\bar{g}(x) = \bar{a}x$. We can obtain $\bar{a}$ using simulation: we just loop over the process, collect the outputs, and the average them as in the following:

```
a_out=[] # output container
for i in range(100000): #100000 loops
    X, y = gen_sindata(2)
    lr.fit(X,y)
    y_pred = lr.predict(xi.reshape(-1,1))
    a_out.append(lr.coef_[0,0])
a_bar = np.mean(a_out) # approx 1.43
```

Bias and Variance

Q: What are the mathematical expressions of bias$(\mathbf{x})$ and var$(\mathbf{x})$ for this example?
A:

$$\text{bias}(\mathbf{x}) = \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2$$

$$\text{var}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]$$

In this example, $\mathbf{x}$ is a one-dimensional tensor (scalar) x:

$$\text{bias}(x) = \left(\bar{g}(x) - f(x)\right)^2 = (\bar{a}x - \sin(\pi x))^2$$

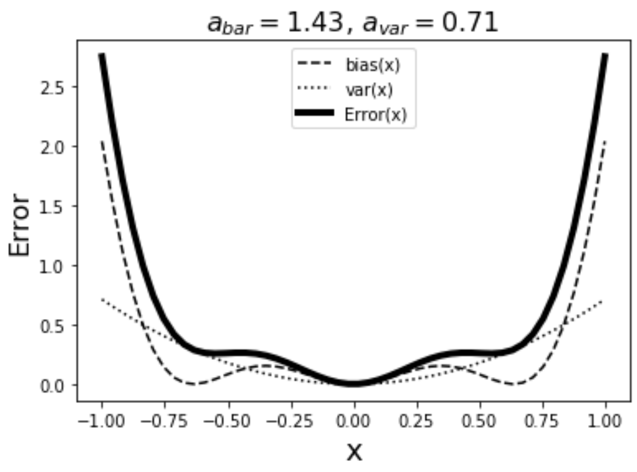
$$\text{var}(x) = \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(x) - \bar{g}(x)\right)^2\right] = \mathbb{E}_{\mathcal{D}}[((a-\bar{a})x)^2] = x^2\mathbb{E}_{\mathcal{D}}[(a-\bar{a})^2] = \text{var}(a)x^2$$

We can then plot these results:

```
a_var = np.var(a_out) # approx 0.71
```

```
xi= np.linspace(-1,1,50)
yi= np.sin(np.pi*xi)

plt.plot(xi,(a_bar*xi-yi)**2,'--k',label='bias(x)')
plt.plot(xi,(a_var)*(xi)**2,':k',label='var(x)')
plt.plot(xi,((a_bar*xi-yi)**2 + a_var*(xi)**2),'-k',lw = 4,
    label='Error(x)')
plt.legend(loc='best')
plt.xlabel('x', fontsize=18)
plt.ylabel('Error',fontsize=16)
plt.title('$a_{bar}=%3.3g$, $a_{var}=%3.3g$'%(a_bar,a_var),fontsize=16)
```



Q: What have you observed?

A:

1. Notice that there is zero bias and zero variance when $x = 0$. This is because the learning method cannot help but get that correct because all the hypotheses happen to match the value of the target function at that point!
2. The errors are worse at the end points. Those points have the most leverage against the hypothesized models and result in the worst errors.

Finally, we can compute the expected out-of-sample error and its `bias` and `var` components by generating a test set and averaging *x* on this new set:

$$\mathbb{E}_{\mathcal{D}}\left[E_{out}\left(g^{(\mathcal{D})}\right)\right] = \mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x}) + \text{var}(\mathbf{x})] = \text{bias} + \text{var}$$

$$\text{bias} = \mathbb{E}_x[\text{bias}(x)] = \mathbb{E}_x[(\bar{a}x - \sin(\pi x))^2]$$
$$\text{var} = \mathbb{E}_x[\text{var}(x)] = \mathbb{E}_x[\text{var}(a)x^2]$$

```
# compute bias and variance
cnt = 1000000
bias = 0
var = 0
```

```
x = np.random.uniform(-1, 1, size = cnt)
for i in range(cnt):
    bias += (a_bar*x[i] - np.sin(np.pi*x[i]))**2
    var += a_var * x[i] * x[i]
bias = bias / cnt # approx 0.268
var = var / cnt # approx 0.236
```

The bias is 0.268 and the variance is 0.236. The total generalization error is 0.504 for $h(x) = ax$.

Food for thought

1. `bias` captures the concept that, even if all possible data were presented to the learning method, it would still differ from the target function by this amount. On the other hand, `variance` shows the variation in the final hypothesis, depending on the training data set, notwithstanding the target function. Thus, the tension between approximation and generalization is captured by these two terms.
2. Unfortunately, the `bias` and `variance` cannot be computed in practice, since they depend on the target function and the input probability distribution (both unknown). Thus, the bias-variance decomposition is a conceptual tool which is helpful when it comes to developing a model.
3. There are two typical strategies to improve the generalization error. The first is to try to lower the `variance` without significantly increasing the `bias`, and the second is to lower the `bias` without significantly increasing the `variance`. Reducing the `bias` without increasing the `variance` requires some prior information regarding the target function $f(\mathbf{x})$ to steer the selection of in the direction of $f(\mathbf{x})$, and this task is largely **application-specific**. On the other hand, reducing the `variance` without compromising the `bias` can be done through general techniques that we will learn later in the course.

You can download the above Python codes `Bias_variance.ipynb` from the course website.