Bias Variance Decomposition

Consider the regression problem where given x we want to predict a real valued outcome y. $\ell(h(x), y) = (h(x) - y)^2$

Eg $\quad x = (\#\text{bedrooms}, \text{ sq footage of house}, \#\text{ bathrooms}, \ldots)$

$\quad y = \text{house price}$

$$\text{Test error}(h) = \underset{(x,y)\sim P}{E}\left[(h(x) - y)^2\right]$$

Bayes Optimal Predictor = hypothesis with smallest possible test loss

$$\bar{y}(x) = E[y \mid x]$$

is the Bayes optimal predictor

$$\underset{(x,y)\sim P}{E}\left[(\bar{y}(x) - y)^2\right] = \text{inherent (unavoidable) noise}$$

No method with any amount of data can beat the above test loss

Say we have an Algorithm that takes as input dataset D and outputs hypothesis h_D. But h_D depends on sample D which is a random draw. We are interested in understanding the expected test error

$$\text{Expected Test Error} = \underset{D\sim P^n}{E}\left[\underset{(x,y)\sim P}{E}\left[(h_D(x) - y)^2\right]\right]$$

Since h_D is a random let us consider its expected behavior:

$$\bar{h} := \underset{D\sim P^n}{E}[h_D] \qquad \text{ie} \qquad \bar{h}(x) = \underset{D\sim P^n}{E}[h_D(x)]$$

Bias : $\mathop{E}_{x\sim P}\left[(\bar{h}(x) - \bar{y}(x))^2 \right]$

Expected sq. distance between expected model of our Algorithm and the best possible model

Variance : $\mathop{E}_{D\sim P^n}\left[\mathop{E}_{x\sim P}\left[(\bar{h}(x) - h_D(x))^2 \right]\right]$

Fluctuation of Algorithm's random model around its mean

We will see that:

Expected test error = Bias + Variance + Inherent noise

$$\text{Expected Test Error} = \mathop{E}_{D\sim P^n}\left[\mathop{E}_{(x,y)\sim P}\left[(h_D(x) - y)^2 \right]\right]$$

$$= \mathop{E}_{D\sim P^n}\left[\mathop{E}_{(x,y)}\left[(\underbrace{h_D(x) - \bar{y}(x)}_{A} + \underbrace{\bar{y}(x) - y}_{B})^2 \right]\right]$$

$$= \mathop{E}_{D\sim P^n}\left[\mathop{E}_{(x,y)}\left[\underbrace{(h_D(x) - \bar{y}(x))^2}_{A^2} + \underbrace{2(h_D(x) - \bar{y}(x))(\bar{y}(x) - y)}_{2AB} + \underbrace{(\bar{y}(x) - y)^2}_{B^2} \right]\right]$$

$$= \mathop{E}_{D\sim P^n} \mathop{E}_{x,y}\left[(h_D(x) - \bar{y}(x))^2 \right] + 2 \underbrace{\mathop{E}_{D\sim P^n} \mathop{E}_{(x,y)}\left[(h_D(x) - \bar{y}(x))(\bar{y}(x) - y) \right]}_{= 0\ ??}$$

$$+ \mathop{E}_{(x,y)}\left[(\bar{y}(x) - y)^2 \right]$$

$$= \mathop{E}_{D\sim P^n} \mathop{E}_{(x,y)}\left[(h_D(x) - \bar{y}(x))^2 \right] + \underbrace{E\left[(\bar{y}(x) - y)^2 \right]}_{\text{"inherent noise"}}$$

$$= \mathop{E}_{D\sim P^n} \mathop{E}_{(x,y)}\left[(\underbrace{h_D(x) - \bar{h}(x)}_{A} + \underbrace{\bar{h}(x) - \bar{y}(x)}_{B})^2 \right] + \underbrace{E\left[(\bar{y}(x) - y)^2 \right]}_{\text{"inherent noise"}}$$

$$= \underset{D}{E}\,\underset{(x,y)}{E}\Big[\underbrace{(h_D(x)-\bar{h}(x))^2}_{A^2} + \underbrace{2(h_D(x)-\bar{h}(x))(\bar{h}(x)-\bar{y}(x))}_{2AB} + \underbrace{(\bar{h}(x)-\bar{y}(x))^2}_{B^2}\Big]$$

$$+ \underbrace{E\big[(\bar{y}(x)-y)^2\big]}_{\text{"inherent noise"}}$$

$$= \underbrace{\underset{D}{E}\,\underset{x}{E}\big[(h_D(x)-\bar{h}(x))^2\big]}_{\text{"Variance"}} + \underbrace{2\,\underset{D}{E}\,\underset{x}{E}\big[(h_D(x)-\bar{h}(x))(\bar{h}(x)-\bar{y}(x))\big]}_{=0\;??}$$

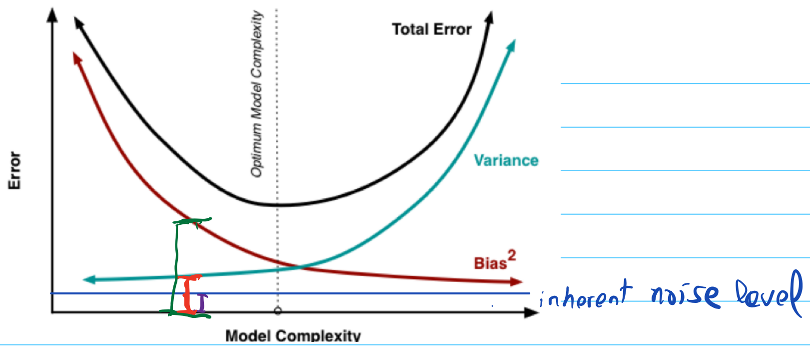
$$+ \underbrace{\underset{x}{E}\big[(\bar{h}(x)-\bar{y}(x))^2\big]}_{\text{"Bias"}} + \underbrace{E\big[(\bar{y}(x)-y)^2\big]}_{\text{"inherent noise"}}$$

$$= \underbrace{\underset{D}{E}\,\underset{x}{E}\big[(h_D(x)-\bar{h}(x))^2\big]}_{\text{"Variance"}} + \underbrace{\underset{x}{E}\big[(\bar{h}(x)-\bar{y}(x))^2\big]}_{\text{"Bias"}} + \underbrace{E\big[(\bar{y}(x)-y)^2\big]}_{\text{"inherent noise"}}$$

Variance: Captures how much your classifier changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set (overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

Bias: What is the inherent error that you obtain from your classifier even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to your model.

Noise: How big is the data-intrinsic noise? This error measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.



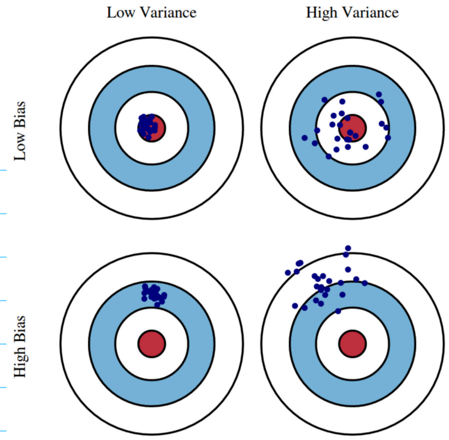


Fig 1: Graphical illustration of bias and variance. (Source http://scott.fortmann-roe.com/docs/BiasVariance.html) Fig 2: The variation of Bias and Variance with the model complexity. This is similar to the concept of overfitting and underfitting. More complex models overfit while the simplest models underfit. (Source http://scott.fortmann-roe.com/docs/BiasVariance.html)

Detecting High Bias and High Variance

If a classifier is under-performing (e.g. if the test or training error is too high), there are several ways to improve performance. To find out which of these many techniques is the right one for the situation, the first step is to determine the root of the problem. The graph above plots the training error and the test error and can be divided into two overarching regimes. In the first regime (on the left side of the graph), training error is below the desired error threshold (denoted by ϵ), but test error is significantly higher. In the second regime (on the right side of the graph), test error is remarkably close to training error, but both are above the desired tolerance of ϵ.

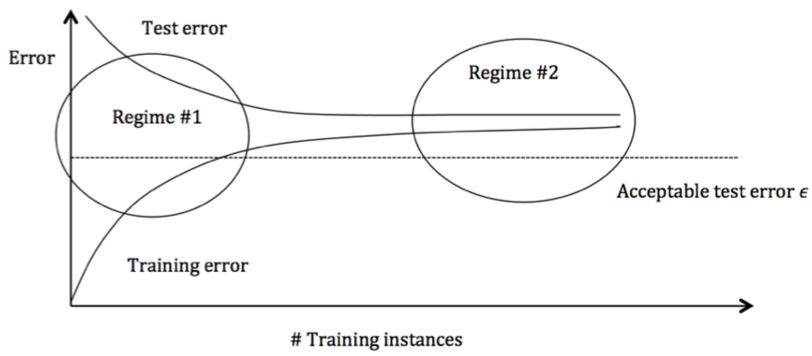


Figure 3: Test and training error as the number of training instances increases.

Regime 1 (High Variance)

In the first regime, the cause of the poor performance is high variance.

Symptoms:

1. Training error is much lower than test error
2. Training error is lower than ϵ
3. Test error is above ϵ

Remedies:

- Add more training data
- Reduce model complexity -- complex models are prone to high variance
- Bagging (will be covered later in the course)

Regime 2 (High Bias)

Unlike the first regime, the second regime indicates high bias: the model being used is not robust enough to produce an accurate prediction.

Symptoms:

1. Training error is higher than ϵ

Remedies:

- Use more complex model (e.g. kernelize, use non-linear models)
- Add features
- Boosting (will be covered later in the course)