Given $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$, drawn iid from some distribution P

Assume a regression setting ($y \in \mathbb{R}$)

Consider that for any given input $\vec{x}$ there might not exist a unique label y.

So for any given feature vector $\vec{x}$, there is a distribution over possible labels.

We thus define the following:

Expected Label (given $\vec{x} \in \mathbb{R}^d$)

$$\bar{y}(\vec{x}) = \mathbb{E}_{y|\vec{x}}[y] = \int_y y \, \mathbb{P}(y|\vec{x}) \, dy$$

This denotes the label you would expect to obtain, given a feature vector $\vec{x}$.

Alright, so we draw our training set D, consisting of n inputs, iid from the distribution P.

As a second step we typically call some machine learning algorithm A on this data set to learn a hypothesis (aka classifier).

Formally, we denote this process as $h_D = A(D)$.

For a **given** h_D, learned on a dataset D w/ algorithm $\mathcal{A}$, we can compute the generalization error as follows:

Expected Test Error (given h_D):

$$\mathbb{E}_{(\vec{x},y)\sim P}\left[(h_d(\vec{x})-y)^2\right] = \int_x \int_y (h_d(\vec{x})-y)^2 \, P(\vec{x},y)\, \partial y\, \partial x$$

Note: one can use other loss functions. Squared loss is shown b/c it's very common and has nice properties.

The previous statement is true for a given training set D itself is drawn from P^n, and is therefore a random variable.

Further, h_D is a function of D, and is therefore also a random variable. We thus compute its expectation.

Expected Classifier (given $\mathcal{A}$):

$$\bar{h} = \mathbb{E}_{D\sim P^n}[h_D] = \int_{\mathcal{D}} h_D \, P(D)\, \partial D$$

where $P(D)$ is the probability of drawing D from P^n.

$\bar{h}$ is a weighted average over all functions.

We can also use the fact that h_D is a random variable to compute the expected test error given only $\mathcal{A}$, taking the expectation also over D.

Expected Test Error (given A):

$$\mathop{\mathbb{E}}_{\substack{(\vec{x},y)\sim P \\ D\sim P^n}}\left[(h_D(\vec{x})-y)^2\right] = \int_D\int_x\int_y (h_D(x)-y)^2 P(\vec{x},y)P(D)\,\partial y\,\partial x\,\partial D$$

To clarify, D is our training points, and the $(\vec{x},y)$ pairs are the test points.

We are interested in **exactly** this expression, because it evaluates the quality of a machine learning algorithm A w/ respect to a data distribution $P(X,Y)$.

This expression decomposes into **THREE** meaningful terms.

Decomposition of Expected Test Error

$$\mathbb{E}_{x,Y,D}\left[(h_D(x)-y)^2\right] = \mathbb{E}_{x,Y,D}\left[\left[(h_D(\vec{x})-\bar{h}(\vec{x}))+(\bar{h}(\vec{x})-y)\right]^2\right]$$

$$= \mathbb{E}_{x,D}\left[(h_D(\vec{x})-\bar{h}(\vec{x}))^2\right] + 2\,\mathbb{E}_{x,Y,D}\left[(h_D(\vec{x})-\bar{h}(\vec{x}))(\bar{h}(\vec{x})-y)\right] + \mathbb{E}_{x,Y}\left[(\bar{h}(\vec{x})-y)^2\right]$$

Let's evaluate middle term

$$\begin{aligned}\mathbb{E}_{x,Y,D}\left[(h_D(\vec{x})-\bar{h}(\vec{x}))(\bar{h}(\vec{x})-y)\right] &= \mathbb{E}_{x,Y}\left[\mathbb{E}_D\left[h_D(\vec{x})-\bar{h}(\vec{x})\right](\bar{h}(\vec{x})-y)\right]\\ &= \mathbb{E}_{x,Y}\left[(\mathbb{E}_D[h_D(\vec{x})]-\mathbb{E}_D[\bar{h}(\vec{x})])(\bar{h}(\vec{x})-y)\right]\\ &= \mathbb{E}_{x,Y}\left[(\bar{h}(x)-\bar{h}(\vec{x}))(\bar{h}(x)-y)\right]\\ &= \mathbb{E}_{x,Y}[0]\\ &= 0\end{aligned}$$

Thus we're left w/ variance and another term

$$\mathbb{E}_{x,Y,D}\left[(h_D(\vec{x})-y)^2\right] = \underbrace{\mathbb{E}_{x,D}\left[(h_D(\vec{x})-\bar{h}(\vec{x}))^2\right]}_{\text{Variance}} + \mathbb{E}_{x,Y}\left[(\bar{h}(\vec{x})-y)^2\right]$$

We break the second term down as follows:

$$\mathbb{E}_{x,Y}\left[(\bar{h}(\vec{x})-y)^2\right] = \mathbb{E}_{x,Y}\left[\left((\bar{h}(\vec{x})-\bar{y}(\vec{x}))+(\bar{y}(\vec{x})-y)\right)^2\right]$$

$$= \underbrace{\mathbb{E}_{x,Y}\left[(\bar{h}(x)-\bar{y}(\vec{x}))^2\right]}_{\text{Bias}} + \underbrace{\mathbb{E}_{x,Y}\left[(\bar{y}(\vec{x})-y)^2\right]}_{\text{Noise}} + \underbrace{2\,\mathbb{E}_{x,Y}\left[(\bar{h}(x)-\bar{y}(x))(\bar{y}(\vec{x})-y)\right]}_{0}$$

To see the third term is zero, observe:

$$\mathbb{E}_{x,Y}\left[(\bar{h}(\vec{x}) - \bar{y}(\vec{x}))(\bar{y}(\vec{x}) - y)\right] = \mathbb{E}_x\left[\mathbb{E}_{y|x}\left[\bar{y}(\vec{x}) - y\right](\bar{h}(\vec{x}) - \bar{y}(\vec{x}))\right]$$

$$= \mathbb{E}_x\left[\left(\mathbb{E}_{y|x}\left[\bar{y}(\vec{x})\right] - \mathbb{E}_{y|x}\left[y\right]\right)(\bar{h}(\vec{x}) - \bar{y}(\vec{x}))\right]$$

$$= \mathbb{E}_x\left[(\bar{y}(x) - \bar{y}(x))(\bar{h}(x) - \bar{y}(\vec{x}))\right]$$

$$= \mathbb{E}_x[0]$$

$$= 0$$

This gives us the decomposition of the test error as follows:

$$\mathbb{E}_{x,Y,D}\left[(h_D(\vec{x}) - y)^2\right] = \underbrace{\mathbb{E}_{x,D}\left[(h_D(\vec{x}) - \bar{h}(\vec{x}))^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x,Y}\left[(\bar{h}(x) - \bar{y}(\vec{x}))^2\right]}_{\text{Bias}} + \underbrace{\mathbb{E}_{x,Y}\left[(\bar{y}(\vec{x}) - y)^2\right]}_{\text{Noise}}$$

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? If we have the best possible 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 w/ infinite training data? This is due to your classifier being biased to a particular kind of solution. In other words, bias is inherent to your model.

Noise: Measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.

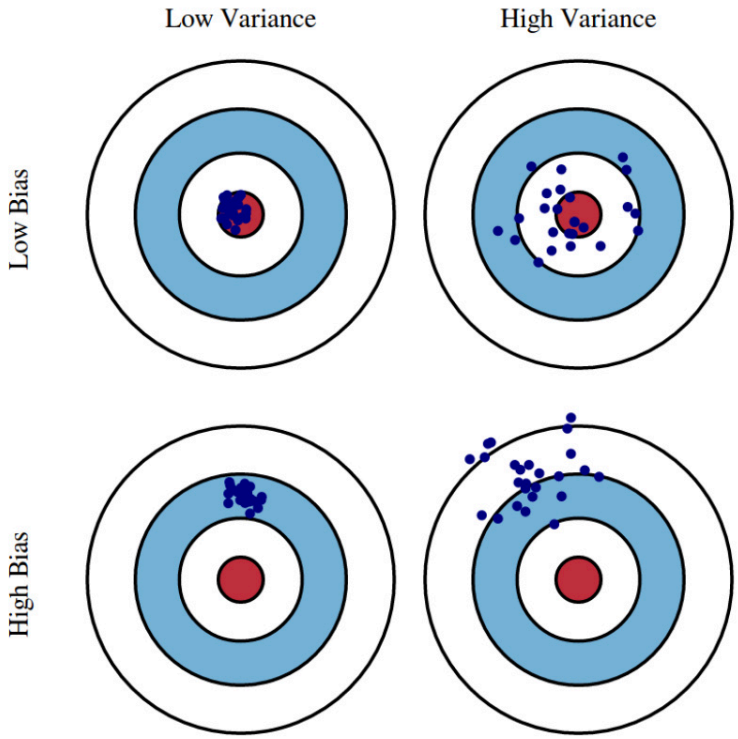

Fig 1: Graphical illustration of bias and variance.
Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

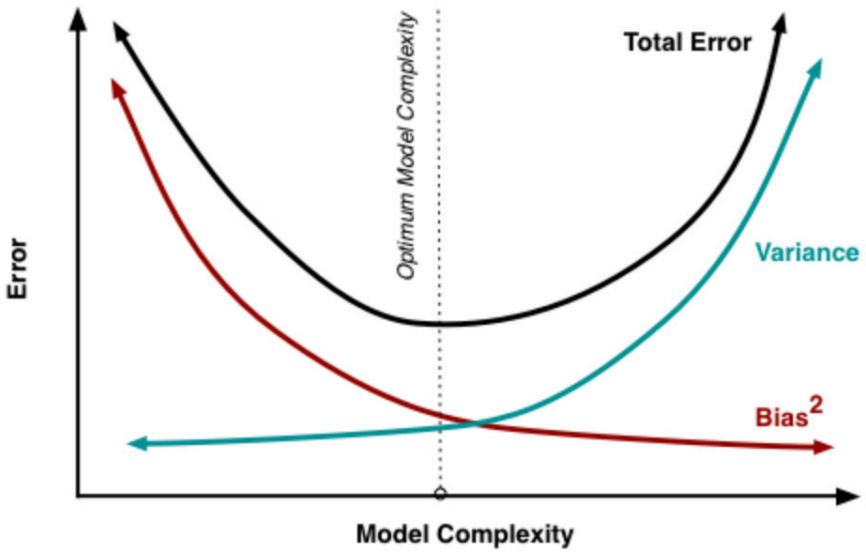

Fig 2: The variation of Bias and Variance with the model complexity. This is similar to the concept of overfitting and underfitting. More complex models overfit while the simplest models underfit.
Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

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.

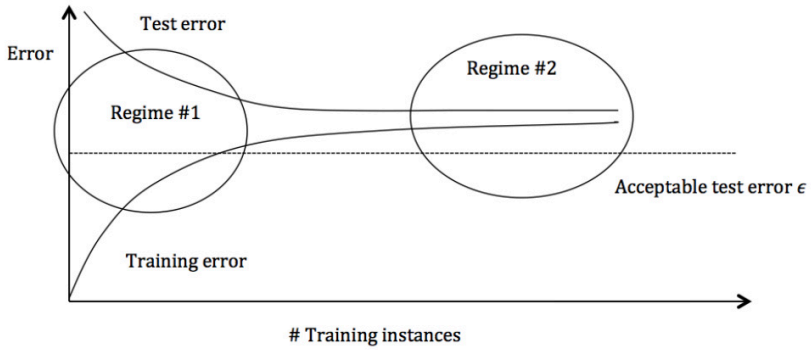

Figure 3: Test and training error as the number of training instances increases.

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.

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)