# Topic 6: Bias-Variance Problem



According to Section 7.5 of `https://bookdown.org/jefftemplewebb/IS-6489/`, **overfitting** is a major hazard in predictive analytics, especially when using machine learning algorithms like random forest (Topic 5) which, without proper tuning, can learn sample data almost perfectly, essentially fitting noise. When such a model is used to predict new data, with different noises, model performance can be shockingly bad. Overfitting error can cause very funny result as pointed out by Dr Kilian Q. Weinberger (`http://kilian.cs.cornell.edu/`) in his license plate recognition example(?) Cornell CS4780 SP17 lecture (`https://www.youtube.com/watch?v=zj-5nkNKAow`).

In this topic, we explore the difference between a predictive model and the true model using the notion of bias-variance tradeoff and the *generalisation error*, which measures how accurately a predictive model is able to predict outcome values for previously unseen data. The generalisation error is related to overfitting issue. Since it is impossible to know the bias-variance and the generalisation errors except for very rare special cases, we need resampling methods to estimate them. Two of the most commonly used resampling methods are the *cross-validation* (Section 6.4.1) and the *bootstrap*.

## 6.1 Training vs Testing Errors

The main purpose of **supervised learning/predictive modelling** is **to find a model with a minimum generalisation error**.

The generalisation error can be theoretically estimated for very limited analytical formulations.

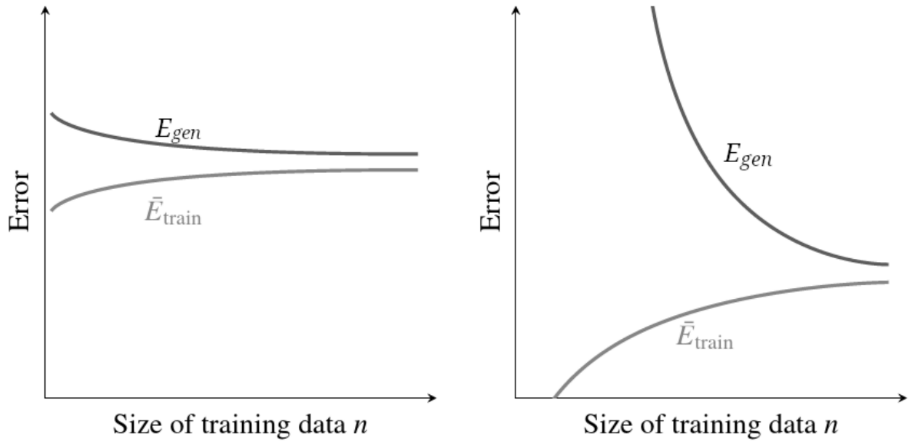
In Topic 1 and Topic 2, the generalisation error $E_{gen}$ is estimated either using (i) holdout method's testing error, (ii) k-fold CV error or (iii) LOOCV (the extreme case of k-fold CV with $k = n$).

There are two things that affects the learning of model from the data:

(a) the number of instances / training samples $n$

(b) the hyperparameter of the model

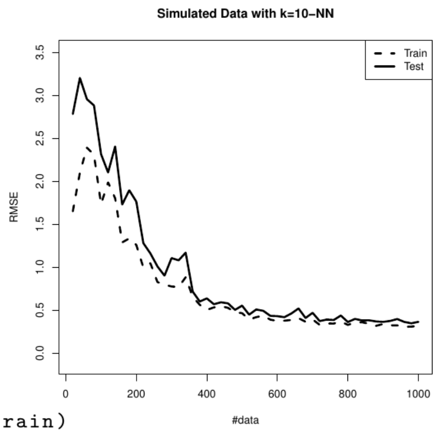
(a) Why are we not using the training error for the estimation of generalisation error? According to Lindholm et al. [2022], the training error and generalisation error has **generalisation gap** which looks roughly like the following figures.




Note that the generalisation error $E_{gen}$ is usually larger than the training error $E_{train}$. So $E_{train}$ may be too optimistic!

The curves are usually not smooth as is shown by our simulation below.

```
library(FNN)    # Fast Nearest Neighbor Search Algorithms and Applications
k = 10          # Using kNN(k=10)

rmse = function(actual, predicted) { sqrt(mean((actual - predicted)^2)) }

make_knn_pred = function(k, training, predicting) {
  pred = FNN::knn.reg(train = training["X"],
                      test = predicting["X"],
                      y = training$y, k = k)$pred
  act  = predicting$y
  rmse(predicted = pred, actual = act)
}

# Simulated Data is based on the function f:
f = function(x) {x*sin(x**2)}

set.seed(2025)

# Nlist = Number of data / instances
Nlist = seq(20,1000,20)
knn_trn_rmse = rep(0,length(Nlist))
knn_tst_rmse = rep(0,length(Nlist))

sig = 0.3
NT = 1000
X.test = runif(NT,min=0,max=2*pi)
y.test = f(X.test) + rnorm(NT, 0, sig)
data.test  = data.frame(X=X.test, y=y.test)

for(i in 1:length(Nlist)) {
  NS = Nlist[i]
  X = runif(NS,min=0,max=2*pi)
  y = f(X) + rnorm(NS, 0, sig)
  data.train = data.frame(X=X, y=y)

  knn_trn_rmse[i] = make_knn_pred(k, data.train, data.train)
  knn_tst_rmse[i] = make_knn_pred(k, data.train, data.test)
}

plot(Nlist, knn_trn_rmse, type='l', lty=2, lwd=3,
  xlab="#data", ylab="RMSE", ylim=c(-0.1,3.5),
  main=paste0("Simulated Data with k=",k,"-NN"))
lines(Nlist, knn_tst_rmse,lwd=3)
legend("topright",c("Train","Test"), col=rep(1,2), lty=2:1, lwd=3)
```




We can conclude that [Lindholm et al., 2022]:

- The generalisation error $E_{gen}$ will, on average, larger than the training error $E_{train}$.

- The generalisation gap $|E_{gen} - E_{train}$ typically decrease as $n$ increases.

Some data may have bias causing the model $h$ with the training error and the test error which are bias with large number of training data as illustrated in Figure 6.1.

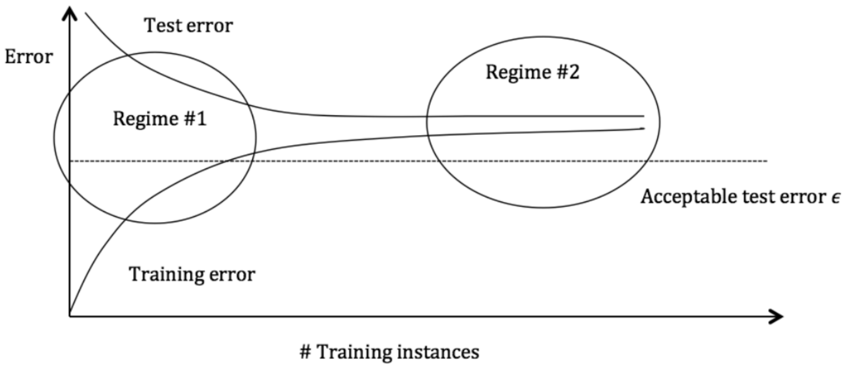



Figure 6.1: Test and training error as the number of training instances increases.

In the first regime (on the left side of the graph), training error is below the desired error threshold (denoted by $\epsilon$), but test error is significantly higher. The cause of the poor performance of $h$ is "high variance".

**Regime 1 (High Variance)**

| Symptoms | Remedies |
|---|---|
| Training error is much lower than test error | Add more training data |
| Training error is lower than $\epsilon$ | Reduce model complexity — complex models are prone to high variance |
| Test error is above $\epsilon$ | Bagging (next topic) |

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 $\epsilon$. This indicates "high bias", i.e. the model being used is not robust enough to produce an accurate prediction.

**Regime 2 (High Bias)**

| Symptoms | Remedies |
|---|---|
| Training error is higher than $\epsilon$ | • Use more complex model (e.g. kernelize, use non-linear models)<br>• Add features<br>• Boosting (next topic) |

The bias-variance analysis in the following sections will be useful in the understanding of underfitting-overfitting problem as well as the development of ensemble methods in the next topic.

## 6.2 Bias-Variance Tradeoff for a Predictive Model

If we have a predictive model $h(x)$ and an true model $f(x) + \epsilon$ (with $\mathbb{E}_{X,\epsilon}[\epsilon] = 0$), the mean square error (MSE) is usually used to measure their difference for the regression problem:

$$MSE = \mathbb{E}_{X,\epsilon}[(h(X) - (f(X) + \epsilon))^2]. \tag{6.1}$$

Note that the true model $f(x)+\epsilon$ have a range and we don't just measure the difference between the predictive model and the true model but we measure the average differences over the range of $x$.

Using probability theory, (6.1) can be expanded to

$$\begin{aligned} MSE &= \mathbb{E}_{X,\epsilon}[(h(X)-f(X))^2 + 2(h(X)-f(X))\epsilon + \epsilon^2] \\ &= \mathbb{E}_X[(h(X)-f(X))^2] + 2\mathbb{E}_X(h(X)-f(X))\mathbb{E}_{X,\epsilon}[\epsilon] + \mathbb{E}_{X,\epsilon}[\epsilon^2] \\ &= \underbrace{\mathbb{E}_X[(h(X)-f(X))^2]}_{\text{bias error}} + \underbrace{\mathrm{Var}[\epsilon]}_{\text{error variance}} . \end{aligned}$$

The bias-variance for a predictive model can be applied to physics or chemistry experiments where $\epsilon$ is the measurement or physical noise and $h(x)$ and $f(x)$ may be the same. It can also be applied to study the expected test error of a trained model $h_D$.

## 6.3 Bias-Variance Tradeoff for Data Samples

In machine learning, the predictive model $h(D, x)$ is usually a function of the training data $D$ and input features $x$, therefore, the bias-variance analysis in the previous section is insufficient.

According to Wikipedia, the bias-variance tradeoff of a predictive model is a property that the variance of the parameter estimated across samples can be reduced by increasing the bias in the estimated parameters. The bias-variance dilemma is the conflict in trying to similtaneously minimise these two sources of error that prevent predictive models from generalising beyond their training set:

- the **bias error** is an error from errorneous assumptions in the predictive model. High bias can cause the predictive model to miss the relevant relations between features and target outputs (underfitting).
- The **variance** is an error from sensitivity to small fluctuations in the training set. High variance may result from the predictive model which models the random noise in the training data (overfitting).

In comparison to the MSE in the previous section, the MSE for predictive models with sampling can be decomposed into three terms — the bias, variance, and irreducible error.

Statiticians assume that the training set $D$, consisting of $n$ inputs, are drawn i.i.d. from some distribution $P$, which is denoted as $D \sim P^n$. The machine learning algorithm $\mathcal{A}$ learns on the data set $D$ a predictive model $h(D, x) = \mathcal{A}(D)$. The MSE can be extended to the *Expected Test Error*:

$$ETE = \mathbb{E}_{\substack{(\boldsymbol{x},y)\sim P \\ D\sim P^n}}\left[(h(D,\boldsymbol{x})-y)^2\right] = \int_D \int_{\boldsymbol{x}} \int_y (h(D,\boldsymbol{x})-y)^2\, \mathbb{P}(\boldsymbol{x},y)\mathbb{P}(D) d\boldsymbol{x} dy dD. \tag{6.2}$$

**Decomposition of Equation 6.2**:

$$\begin{aligned} ETE =& \mathbb{E}_{\boldsymbol{x},y,D}\left[\left[\left(h(D,\boldsymbol{x})-\overline{h}(\boldsymbol{x})\right)+\left(\overline{h}(\boldsymbol{x})-y\right)\right]^2\right] \\ =& \mathbb{E}_{\boldsymbol{x},D}\left[(h(D,\boldsymbol{x})-\overline{h}(\boldsymbol{x}))^2\right] + 2\,\mathbb{E}_{\boldsymbol{x},y,D}\left[\left(h(D,\boldsymbol{x})-\overline{h}(\boldsymbol{x})\right)\left(\overline{h}(\boldsymbol{x})-y\right)\right] + \mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-y\right)^2\right]. \end{aligned} \tag{6.3}$$

The middle term of the above equation is 0 as we show below

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

Note that $\overline{h}$ is a weighted average of trained functions on various data $D$.

$$\overline{h}(\boldsymbol{x}) = \mathbb{E}_{D\sim P^n}[h(D,\boldsymbol{x})] = \int\limits_D h(D,\boldsymbol{x})\mathbb{P}(D)dD$$

where $\mathbb{P}(D)$ is the probability of drawing $D$ from $P^n$.

Returning to the earlier expression (6.3), we're left with the variance and another term

$$\mathbb{E}_{\boldsymbol{x},y,D}\left[\left(h(D,\boldsymbol{x})-y\right)^2\right] = \underbrace{\mathbb{E}_{\boldsymbol{x},D}\left[\left(h(D,\boldsymbol{x})-\overline{h}(\boldsymbol{x})\right)^2\right]}_{\text{Variance}} + \mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-y\right)^2\right] \tag{6.4}$$

Let the expected label $\overline{y}(\boldsymbol{x}) = \mathbb{E}_{y|\boldsymbol{x}}[y(\boldsymbol{x})] = \int\limits_y y\,\mathbb{P}(y|\boldsymbol{x})dy$. We can break down the second term in the equation (6.4) as follows:

$$\mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-y\right)^2\right] = \mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)+\left(\overline{y}(\boldsymbol{x})-y\right)^2\right] \tag{6.5}$$

$$= \underbrace{\mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{y}(\boldsymbol{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{\mathbb{E}_{\boldsymbol{x}}\left[\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)^2\right]}_{\text{Bias}^2} + 2\,\mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\left(\overline{y}(\boldsymbol{x})-y\right)\right] \tag{6.6}$$

The third term in the equation above is 0, as we show below

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\left(\overline{y}(\boldsymbol{x})-y\right)\right] &= \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{y|\boldsymbol{x}}\left[\overline{y}(\boldsymbol{x})-y\right]\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\right]\\ &= \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{y|\boldsymbol{x}}\left[\overline{y}(\boldsymbol{x})-y\right]\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\right]\\ &= \mathbb{E}_{\boldsymbol{x}}\left[\left(\overline{y}(\boldsymbol{x})-\mathbb{E}_{y|\boldsymbol{x}}\left[y\right]\right)\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\right]\\ &= \mathbb{E}_{\boldsymbol{x}}\left[\left(\overline{y}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)\right] = \mathbb{E}_{\boldsymbol{x}}\left[0\right] = 0.\end{aligned}$$

This gives us the decomposition of expected test error (6.2) as follows

$$\boxed{\underbrace{\mathbb{E}_{\boldsymbol{x},y,D}\left[\left(h(D,\boldsymbol{x})-y\right)^2\right]}_{\text{ExpectedTestError}} = \underbrace{\mathbb{E}_{\boldsymbol{x},D}\left[\left(h(D,\boldsymbol{x})-\overline{h}(\boldsymbol{x})\right)^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{\boldsymbol{x},y}\left[\left(\overline{y}(\boldsymbol{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{\mathbb{E}_{\boldsymbol{x}}\left[\left(\overline{h}(\boldsymbol{x})-\overline{y}(\boldsymbol{x})\right)^2\right]}_{\text{Bias}^2}.} \tag{6.7}$$

The "Variance" captures how much the classifier changes if we train on a different training set. How "over-specialised" is the classifier to a particular training set (i.e. is it overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

The data-intrinsic "noise" measures the ambiguity due to our data distribution and feature representation. We can never beat this, it is an aspect of the data.

The inherent error that we obtain from the classifier even with infinite training data is captured by the "bias", which is due to the classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, "bias" is inherent to our model.

**Remark 6.3.1.** The reason to limit ourselves to regression problems is due to the many shortcomings of bias-variance decomposition theory for 0-1 loss function according to `http://rasbt.github.io/mlxtend/user_guide/evaluate/bias_variance_decomp/`.

**Example 6.3.2.** Consider a similar problem to Example 1.7.6 below:

$$y = \sin(x) + R, \quad -\pi < x < 2\pi, \quad R \sim Normal(0, 0.3^2).$$

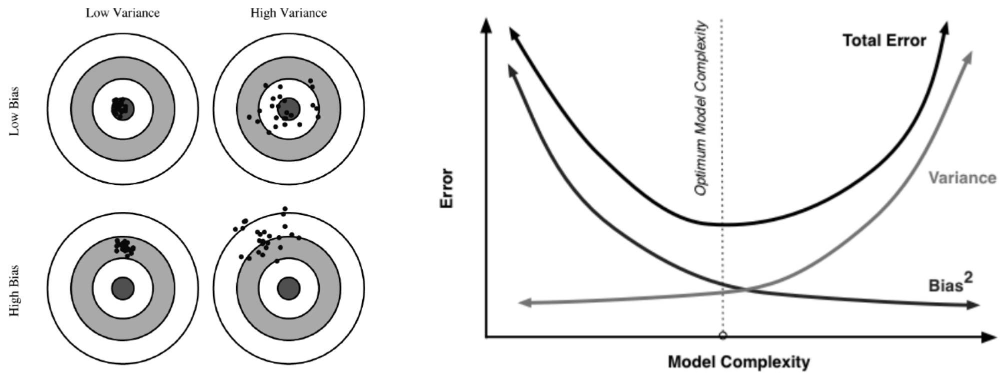
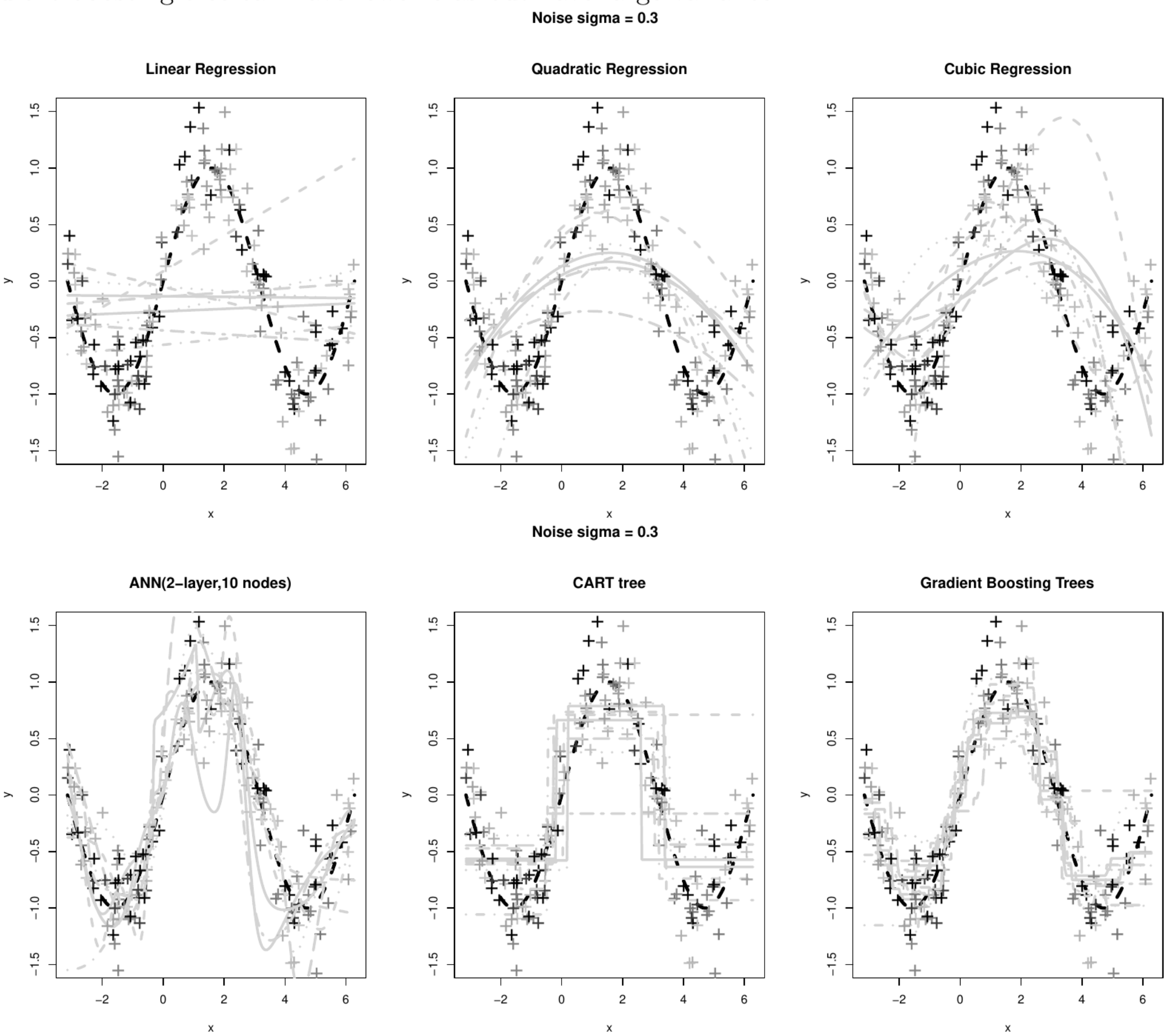



Figure 6.2: (a) Graphical illustration of bias and variance with the output $y$ being a 2D vector. (b) The variation of Bias and Variance with the model complexity. This is similar to the concept of overfitting and underfitting. More complex models overfit while the simplest models underfit. Source: `http://scott.fortmann-roe.com/docs/BiasVariance.html`

By using nine data with 20 samples $D_i,\ i = 1, ..., 9$, we train the predictive models linear regression to gradient boosting trees and we can see the less flexible models like linear regression, quadratic regression, etc. have large bias while the flexible models like ANN, CART tree and gradient boosting tree can have lower bias but have large variance.




□

## 6.4 Resampling Methods

Resampling methods are statistical techniques for estimating statistics from samples. In particular, they can be used to estimate the generalisation error for predictive models. Two most popular resampling methods are the cross-validation approach and the bootstrap approach.

### 6.4.1 Cross-Validation

Suppose the data $D$ is used to train a supervised learning model $\widehat{h}_D$ and suppose that we have a new test data

$$(\boldsymbol{x}_1^{new}, y_1^{new}), \cdots, (\boldsymbol{x}_L^{new}, y_L^{new}) \sim P. \tag{6.8}$$

The general concept associated with "test error" is the *loss function* $\ell$ of individual test data $(\boldsymbol{x}_i^{new}, y_i^{new})$ and the average of loss function — *cost function*[Lindholm et al., 2022], i.e.

$$\frac{1}{L}\sum_{i=1}^{L} \ell(y_i^{new}, \widehat{h}_D(\boldsymbol{x}_i^{new})) \to \mathbb{E}[\ell(Y^{new}, \widehat{h}_D(X^{new}))] \tag{6.9}$$

where the right is the *theoretical test set error* for a predictive model training on a particular training set $\widehat{h}_D$. The *theoretical generalisation error* is defined as the average of theoretical test set error over all possible training sets, i.e.

$$\mathbb{E}_D\left[\mathbb{E}_{(X^{new},Y^{new})}[\ell(Y^{new}, \widehat{h}_D(X^{new}))]\right]. \tag{6.10}$$

For classification problems, the usual form of test error (6.9) and $\ell$ is

$$\frac{1}{L}\sum_{i=1}^{L} I(y_i^{new} \neq \widehat{h}_D(\boldsymbol{x}_i^{new})).$$

For regression problems, the usual form of test error (6.9) and $\ell$ is

$$\frac{1}{L}\sum_{i=1}^{L} (y_i^{new} - \widehat{h}_D(\boldsymbol{x}_i^{new}))^2.$$

A true test data (6.8) is not available at the time we want to run and tune our supervised learning algorithm. Therefore, we need artificial ways to construct training and testing datasets:

1. Holdout method/Split-test validation (Section 1.8.5): shuffle the data and split into two sets (train-test) or three sets (train-validate-test for hyperparameter selection). The performance measure obtained is dependent on the sampling/shuffling;
2. Random subsampling: average out multiple holdout method;
3. K-fold cross-validation (CV) and Leave-one-out cross-validation (LOOCV) (Section 2.9): the shuffle data is splitted into $k$ equal partitions (when $k = n$, the CV is called LOOCV) and use each partition as test data and the remainder as training data. From this, we can get $k$ performance metrics to estimate theoretical generalisation error which would usually be less bias compare to the holdout method.

In holdout method, the estimation of the generalisation error (6.10) is

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i^{new}, \widehat{h}_D(\boldsymbol{x}_i^{new})).$$

To estimate a confidence interval for accuracy in holdout method, we apply the binomial experiment and the normal distribution approximation. The confidence interval of accuracy,

$acc = X/n$ with mean $p$ and variance $p(1-p)/n$ follows:

$$P\left(-Z_{\alpha/2} \leq \frac{acc - p}{\sqrt{p(1-p)/n}} \leq Z_{1-\alpha/2}\right) = 1 - \alpha$$

where $-Z_{\alpha/2}$ and $Z_{1-\alpha/2}$ are the lower and upper bounds are related a standard normal distribution at a confidence level $1-\alpha$.

Consider a model with an accuracy of 80% when evaluated on 100 test records. For a confidence level of $\alpha = 0.95$, $Z_{\alpha/2} = Z_{1-\alpha/2} = 1.96$ and the confidence interval is (Check out the Wilson score interval in Wikipedia:Binomial proportional confidence interval):

$$\begin{aligned}&\frac{2\times n\times acc + Z_{\alpha/2}^2 \pm Z_{\alpha/2}\sqrt{Z_{\alpha/2}^2 + 4n\,(acc - acc^2)}}{2(n+Z_{\alpha/2}^2)}\\ =&\frac{2\times 100\times 0.8 + 1.96^2 \pm 1.96\sqrt{1.96^2 + 4\times 100\times (0.8-0.8^2)}}{2(100+1.96^2)} = [0.7112, 0.8666]\end{aligned}$$

However, the caret's confusionMatrix uses the Clopper-Pearson interval which is

$$\left(1 + \frac{n-x+1}{(x)F[\frac{\alpha}{2}; 2(x), 2(n-x+1)]}\right)^{-1} < p < \left(1 + \frac{n-x}{(x+1)F[1-\frac{\alpha}{2}; 2(x+1), 2(n-x)]}\right)^{-1}$$

for $x = n\times acc$ is the number of success, $x \neq 0$ and $x \neq n$. When $x = 0$, the interval is $(0, 1-(\frac{\alpha}{2})^{\frac{1}{n}})$; When $x = n$, the interval is $((\frac{\alpha}{2})^{\frac{1}{n}}, 1)$.

To compare the performance of two models $M_1$ and $M_2$, we suppose they are evaluated on two independent test sets, $D_1$ and $D_2$. Suppose the error rate for $M_1$ on $D_1$ is $e_1$ and the error rate for $M_2$ on $D_2$ is $e_2$. If $|D_1|$ and $|D_2|$ are sufficiently large, $e_1$, $e_2$ follows normal distributions. Then $d = e_1 - e_2$ also follows normal distribution and the $(1-\alpha)$ confidence interval is

$$|e_1 - e_2| \pm Z_{\alpha/2}\sqrt{\frac{e_1(1-e_1)}{|D_1|} + \frac{e_2(1-e_2)}{|D_2|}}.$$

Random subsampling (or Monte Carlo crossvalidation according to R.R. Picard and R.D. Cook, J. Am. Stat. Assoc., 79 (1984) 575–583) repeat the holdout method sevaral times (with different random samples) to improve the estimation of the classifier's performance. In contrast to a full CV method, random subsampling has been shown to be asymptotically consistent (J. Shao, J. Am. Stat. Assoc., 88 (1993) 486–494.) resulting in more pessimistic predictions of the test data compared with CV.

In k-fold CV, suppose that the data $D$ are randomly partition their indices $\{1,\cdots,n\}$ into $\{B_k\}$ such that $\bigcup_{k=1}^K B_k = \{1,\cdots,n\}$ and $B_j \cap B_k = \emptyset$ $(j \neq k)$. If a partition with $B_k$ removed is used to produce a training model $\widehat{h}_{n-|B_k|}^{-B_k}$. The estimation of the generalisation error (6.10) is

$$\mathbb{E}_D\left[\mathbb{E}_{(X^{new},Y^{new})}[\ell(Y^{new}, \widehat{h}_{n-|B_k|}^{(-B_k)}(X^{new}))]\right] \approx \frac{1}{K}\sum_{i=1}^K \frac{1}{|B_k|}\sum_{i\in B_k} \ell(y_i^{new}, \widehat{h}_{n-|B_k|}^{(-B_k)}(\boldsymbol{x}_i^{new})).$$

In LOOCV, we used the $i$th sample point as test data and the remaining $n-1$ sample points as training data to train a predictive model $\widehat{h}_{n-1}^{(-i)}$. The estimation of the generalisation error (6.10) is

$$\mathbb{E}_D\left[\mathbb{E}_{(X^{new},Y^{new})}[\ell(Y^{new},\widehat{h}_{n-1}^{(-i)}(X^{new}))]\right]\approx\frac{1}{n}\sum_{i=1}^{n}\ell(y_i^{new},\widehat{h}_{n-1}^{(-i)}(\boldsymbol{x}_i^{new})).$$

Since training sample $D$ is close to $n$, the bias is small. The variance is usually high because $n$ training sets are similar to each other due to strong correlation:

$$\mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\ell(y_i^{new},\widehat{h}_{n-1}^{(-i)}(\boldsymbol{x}_i^{new}))\right)=\frac{1}{n^2}\underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cov}\left(\ell(y_i^{new},\widehat{h}_{n-1}^{(-i)}(\boldsymbol{x}_i^{new})),\ell(y_i^{new},\widehat{h}_{n-1}^{(-i)}(\boldsymbol{x}_i^{new}))\right)}_{\text{may be large due to correlation}}.$$

Note that this is not true in general as P. Burman (1989)'s paper "A comparitive study of ordinary cross-validation, $v$-fold cross-validation and the repeated learing-testing methods" proved that for least squares linear regression, LOOCV has the smallest asymptotic bias and variance.

### 6.4.2 Bootstrap Approach

According to `https://en.wikipedia.org/wiki/Bootstrapping_(statistics)`, *bootstrapping* is any test or metric that uses **random sampling with replacement**. The bootstrap is a method for estimating the variance of an estimator and for finding approximate confidence intervals for parameters.

Let $X_1,...,X_n\sim P$. Recall that the empirical distribution $P_n$ is defined by

$$P_n(A)=\frac{1}{n}\sum_{i=1}^{n}I(X_i\in A).$$

In other words, $P_n$ puts mass $1/n$ at each $X_i$. A parameter of the form $\theta=T(P)$ is called a *statistical functionl* and that the plug-in estimator is $\hat{\theta}_n=T(P_n)$.

An iid sample of size $n$ drawn from $P_n$ is called a *bootstrap* sample, denoted by

$$X_1^*,...,X_n^*\sim P$$

Bootstrap samples play an important role in what follows. Note that drawing an iid sample $X_1^*,...,X_n^*$ from $P_n$ is equivalent to drawing $n$ observations, with replacement, from the original data $\{X_1,...,X_n\}$. Thus, bootstrap sampling is often described as "resampling the data".

Let $\hat{\theta}_n=g(X_1,...,X_n)$ denote some estimator. We would like to find the variance of $\hat{\theta}_n$. Let

$$\mathrm{Var}_P(\hat{\theta}_n)=\mathrm{Var}_P(g(X_1,...,X_n))=:S_n(P).$$

Let $s^2$ be the sample variance of $\hat{\theta}_n^{(1)},...,\hat{\theta}_n^{(B)}$. By law of large numbers,

$$s^2=\frac{1}{B}\sum_{j=1}^{n}(\hat{\theta}_n^{(j)})^2-\left(\sum_{j=1}^{n}\hat{\theta}_n^{(j)}\right)^2\xrightarrow{P}\mathbb{E}[\hat{\theta}_n^2]-\mathbb{E}[\hat{\theta}_n]^2=\mathrm{Var}_P(\hat{\theta}_n).$$

Since we can take $B$ as large as we want, we have that $s^2\approx\mathrm{Var}\,P(\hat{\theta}_n)$. In other words, we can **approximate $S_n(P)$ by repeatedly simulating $n$ observations from $P$.**

But we don't know $P$. So we estimate $S_n(P)$ with $S_n(P_n)$ where $P_n$ is the empirical distribution. Since $P_n$ is a *consistent estimator*, we expect that $S_n(P_n)\approx S_n(P)$. In other words, bootstrap approximation of the variance, Estimate $S_n(P)$ with $S_n(P_n)$, i.e. $\widehat{\mathrm{Var}_P(\hat{\theta}_n)} = \mathrm{Var}_{P_n}(\hat{\theta}_n)$.

The steps for the Bootstrap Variance Estimator algorithm are

1. Draw a bootstrap sample $X_1^*, ..., X_n^* \sim P_n$. Compute $\hat{\theta}_n^* = g(X_1^*, ..., X_n^*)$.
2. Repeat the previous step, $B$ times, yielding estimators $\hat{\theta}_{n,1}^*, ..., \hat{\theta}_{n,B}^*$.
3. Compute (using population variance rather than sample variance as in James et al. [2013, SS5.2]):

$$\hat{s} = \sqrt{\frac{1}{B}\sum_{j=1}^{B}(\hat{\theta}_{n,j}^* - \overline{\theta})^2}$$

   where $\overline{\theta} = \frac{1}{B}\sum_{j=1}^{B}\hat{\theta}_{n,j}^*$.
4. Output $\hat{s}$.

A very simple illustration is given in Practical 2 to estimate the standard deviation of the population.