Introduction to Machine Learning

Bias and variance analysis

Arthur GRAILLET s182019
Robin FONBONNE s182200
Thomas ROTHEUDT s191895

Academic year 2024-2025

1 Analytical derivations

1. We have the expected generalization error of the k-Nearest Neighbours algorithm:

$$E_{LS}\{E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\}\}$$

We can substitude y by $f(\mathbf{x}) + \epsilon$ and expand the square in the expected squared error written as

$$E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\}$$

to obtain:

$$E_{y|\mathbf{x}}\{(f(\mathbf{x}) - \hat{y}(\mathbf{x}; LS, k))^2 + 2\epsilon(f(\mathbf{x}) - \hat{y}(\mathbf{x}; LS, k)) + \epsilon^2\}$$

Since $E[\epsilon] = 0$ and ϵ is independent of $f(x)$ and $\hat{y}(\mathbf{x}; LS, k)$ the cross-term vanishes and the formula becomes:

$$E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\} = (f(\mathbf{x}) - \hat{y}(\mathbf{x}; LS, k))^2 + E[\epsilon^2]$$

Since $\epsilon \sim \mathcal{N}(0, \sigma^2)$, we have $E[\epsilon^2] = \sigma^2$ it can be written as:

$$(f(\mathbf{x}) - \hat{y}(\mathbf{x}; LS, k))^2 + \sigma^2$$

We can rewrite $\hat{y}(\mathbf{x}; LS, k)$ as the average of the function values at the k-nearest neighbors:

$$\hat{y}(\mathbf{x}; LS, k) = \frac{1}{k}\sum_{l=1}^{k} y_{(l)}$$

where $y_{(l)} = f(\mathbf{x}_{(l)}) + \epsilon$ (with $\epsilon \sim \mathcal{N}(0, \sigma^2)$). Therefore it can be written as:

$$\hat{y}(\mathbf{x}; LS, k) = \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)}) + \epsilon = \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)}) + \frac{1}{k}\sum_{l=1}^{k} \epsilon$$

The expected squared error can be decomposed as follows:

$$\sigma^2 + \left(f(\mathbf{x}) - \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)})\right)^2 + 2\left(f(\mathbf{x}) - \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)})\right)\left(\frac{1}{k}\sum_{l=1}^{k} \epsilon\right) + \left(\frac{1}{k}\sum_{l=1}^{k} \epsilon\right)^2$$

Since ϵ is an independent random variable with a mean of zero, the cross-term has an expectation of zero. We can write the expected generalization error as:

$$E_{LS}\{E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\}\} = E_{LS}\left(\sigma^2\right) + E_{LS}\left[\left(f(\mathbf{x}) - \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)})\right)^2\right] + E_{LS}\left[\left(\frac{1}{k}\sum_{l=1}^{k} \epsilon\right)^2\right]$$

The expectation of a constant is the constant itself and the expectation of the third term

$$E_{LS}\left[\left(\frac{1}{k}\sum_{l=1}^{k} \epsilon\right)^2\right] = \frac{\sigma^2}{k}$$

because $\mathrm{Var}(\epsilon) = \sigma^2$.

Since $\mathbf{x}$ is fixed, the second term is also fixed. Taking the expectation E_{LS} over this term has no effect.

Therefore we can conclude that

$$E_{LS}\{E_{y|\mathbf{x}}\{(y - \hat{y}(\mathbf{x}; LS, k))^2\}\} = \sigma^2 + \left[f(\mathbf{x}) - \frac{1}{k}\sum_{l=1}^{k} f(\mathbf{x}_{(l)})\right]^2 + \frac{\sigma^2}{k}$$

The first term represents the noise, the second the bias, and the third the variance.

2. We know that $f(x) = x^2$. Since we have to evaluate the bias and variance at $x = 0$ we know that $f(x) = 0$.

   The bias can be written as:

$$\text{bias} = \left(\frac{1}{k} \sum_{l=1}^{k} (x_{(l)})^2 \right)^2$$

   Since the training inputs are symmetrically distributed around $x = 0$ on a uniform grid in [-1,+1], the k-neighbors of $x = 0$ include the point $x = 0$, k' positive points, and k' negative points. We have that

$$\sum_{l=1}^{k} (x_{(l)})^2 = 2 \sum_{l=1}^{k'} \left(\frac{i}{N'} \right)^2 = \frac{2}{(N')^2} \sum_{l=1}^{k'} i^2$$

   We can use the formula to write the sum as a function of k':

$$\frac{2}{(N')^2} \sum_{l=1}^{k'} i^2 = \frac{2}{(N')^2} \frac{k'(k'+1)(2k'+1)}{6}$$

   The result must be expressed as a function of k and N, we can replace k' and N' with the relation $k' = \frac{k-1}{2}$ and $N' = \frac{N-1}{2}$:

$$\frac{2}{\frac{(N-1)^2}{4}} \frac{(\frac{k-1}{2})(\frac{k-1}{2}+1)(k-1+1)}{6} = \frac{4k(k-1)(\frac{k-1}{2}+1)}{3(N-1)^2}$$

   The bias can be written as a function of k and N:

$$\text{bias} = \left(\frac{1}{k} \frac{4k(k-1)(\frac{k-1}{2}+1)}{3(N-1)^2} \right)^2 = \left(\frac{4(k-1)(\frac{k-1}{2}+1)}{3(N-1)^2} \right)^2$$

   The variance is already expressed as a function of σ and k:

$$\text{variance} = \frac{\sigma^2}{k}$$

3. - k appears in the formula of the variance and it is obvious that a greater k leads to a smaller variance. But increasing k may lead to an increase of the biais because we would consider more distant points which reduce the flexibility of the model.
   - Increasing the size N of the learning sample generally leads to a denser distribution, therefore with the same k and σ, the k-nearest neighbors around x will be closer to x and we can expect that they better represent the local behavior of $f(x)$. We can say that larger N leads to smaller bias. If we consider the problem explain in 2, we can see that the size N is in the bias formula and increasing N reduces the bias.

     N has no direct impact on the variance but we can note that a larger N allows a larger k which leads to a lower variance.
   - A greater σ means more noise and as we may expect it increases the variance but does not impact the biais.

4. As suggested we'll compute the minimum by running actual experiments. We have to minimize this function:

$$f(k) = \frac{4}{3(N-1)^2} \left(\frac{(k-1)^2}{2} + (k-1) \right)^2 + \frac{\sigma^2}{k}$$

   By testing every possible value of k such that $k = 2k' + 1$ with $0 \leq k' \leq \frac{N-1}{2}$ for each combination of N and σ we have the following results:

	$\sigma = 0$	$\sigma = 0.1$	$\sigma = 0.2$
$N = 25$	1	1	1
$N = 50$	1	1	3

Table 1: Table of k^* considering only odd values for k

The cells represent the k^* for each combination.

If we can still consider even values for k with the formula obtained considering only odd values of k we would have:

	$\sigma = 0$	$\sigma = 0.1$	$\sigma = 0.2$
$N = 25$	1	1	2
$N = 50$	1	2	2

Table 2: Table of k^* considering every value for k

5. Increasing the size of N or σ tends to increase the value of k^*. If $\sigma = 0$ then modifying N won't change k^* because the best value would always be to consider only one element.

2 Empirical analysis

The residual error represents the minimal attainable error, which is the irreducible noise in the data. The key challenge is that we don't know the true function $f^*(x)$ that maps wine characteristics to quality scores. Without this true function $f^*(x)$, we cannot measure how much of the prediction error comes from noise versus other sources of error.

This is because the residual error is defined as the difference between the actual observed values y and the true function values

$$f^*(x) : \mathbb{E}[(y - f^*(x))^2]$$

Even if we build a very good model, we cannot know how close it is to this true underlying function f*(x), and therefore cannot isolate how much of the error is truly irreducible (residual error) versus due to model imperfections.

In our case, this is particularly challenging because the relationship between wine characteristics and quality involves human judgments that do not follow any consistent mathematical function, making the true $f^*(x)$ even more difficult to determine.

Additionally, the observed error is a combination of bias, variance, and residual error.

$$TotalError = Bias + Variance + ResidualError$$

Since bias and variance arise from model imperfections, separating these components from the irreducible residual error becomes nearly impossible without further assumptions or prior knowledge of the noise distribution.

Our protocol consists of the following steps :

Step 1

1. From the dataset P of size N_S, randomly divide the data into multiple subsets of fixed size N (learning samples, LS).
2. Select a fixed, independent test set TS for evaluating the model performance. The test set must remain unchanged and independent of the learning samples.

Step 2

1. For each learning sample LS_j:
   (a) Train a model $\hat{f}_{LS_j}$ using the learning sample LS_j.
   (b) Generate predictions on the test set TS for each trained model.
   (c) Record the predictions for each point $x_i \in TS$.

Step 3 For every point $x_i \in TS$, decompose the total error into variance, bias, and residual error:

1. **Expected Error (Total Error):**

$$\text{Total Error} = \frac{1}{M} \sum_{j=1}^{M} (y_i - \hat{f}_{LS_j}(x_i))^2$$

   where M is the number of models trained using different learning samples LS_j, and y_i is the true value for x_i.

2. **Variance:**

$$\text{Variance} = \frac{1}{M} \sum_{j=1}^{M} (\hat{f}_{LS_j}(x_i) - \bar{\hat{f}}(x_i))^2$$

   where $\bar{\hat{f}}(x_i) = \frac{1}{M} \sum_{j=1}^{M} \hat{f}_{LS_j}(x_i)$ is the mean of the predictions across different learning samples.

3. **Sum of Bias and Residual Error:**

$$\text{Bias} + \text{Residual Error} = \text{Total Error} - \text{Variance}$$

   Since the residual error is constant across different models, any variation in the total error after subtracting variance can be attributed to the bias and residual error.

Step 4

1. For each point $x_i \in TS$, calculate the average values of the variance, bias + residual error, and total error across all the models.

2. Use these average values as the overall estimates for variance, bias + residual error, and total error.

This protocol can be repeated with different hyperparameter settings of the learning algorithm to assess their impact on bias and variance. Since the residual error remains constant, any changes in the decomposed values can be attributed to changes in either bias or variance, allowing us to understand how different hyperparameter choices affect the bias-variance tradeoff.