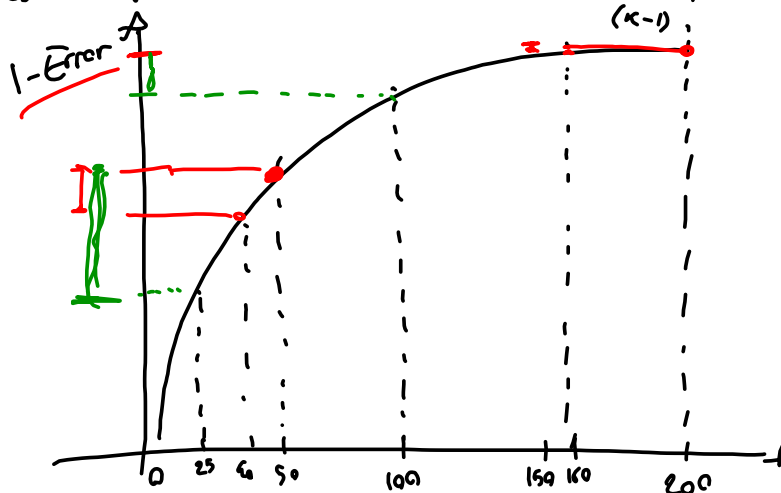


BOOTSTRAPPING

- a disadvantage of cross-validation is that we train the model on less data points than those available $n_{\text{train}} < n$



$$k=5 \quad n=200$$

$$n_{\text{train}} = \frac{200 \times 4}{5} = 160$$

$$n=50$$

$$n_{\text{train}} = \frac{50 \times 4}{5} = 40$$

$$k=2 \quad n=200 \rightarrow n_{\text{train}} = 100$$

$$n=50 \rightarrow n_{\text{train}} = 25$$

Alternative to cross-validation: bootstrapping

- Based on non-parametric bootstrap:

- 1) generate a bootstrap sample by sampling with replacement n observations from the original sample

$$\text{e.g. } Z = \{(x_1, y_1); (x_2, y_2); (x_3, y_3); (x_4, y_4); (x_5, y_5)\} \quad n_{\text{sample}} = 5$$

$$Z_b^* = \{(x_2, y_2); (x_1, y_2); (x_3, y_3); (x_5, y_5); (x_5, y_5)\} \quad n_{\text{sample}}^{\text{bootstrap}} = 5$$

- 2) fit the model on the bootstrap sample $\rightarrow \hat{f}_b^*(x)$

- 3) use the observations not selected at step 1 as a test set, we compute the error on the observations that do not belong to the bootstrap sample

$$\text{e.g. } \text{err}_b^* = \frac{1}{2} \left[(y_1 - \hat{f}_b^*(x_1))^2 + (y_4 - \hat{f}_b^*(x_4))^2 \right]$$

- 4) repeat steps 1 to 3 for $b=1, \dots, B$ with B large, and average the results

$$\hat{\text{err}}^{(1)} \rightarrow \text{err}_{\text{bootstrap}} = \frac{1}{B} \sum_{b=1}^B \text{err}_b^*$$

IMPORTANT: at step 3, we need to use ONLY the observations not selected at point 1

$$\hat{Err}_{best} \rightarrow Err_3^* \neq \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}_3^*(x_i))^2$$

Example:

- classification problem, 2 classes, ^{equal size} response and covariates are independent
no information on the covariates $\rightarrow$ error = 0.5
- simple model: we assign to the observation the class of closest point
(1 - nearest-neighbour)

0	1	0
0	0	1
1	0	0

using the wrong formula

- if $y_i \in Z_3^* \rightarrow$ error = 0

because the closest observation is the observation itself

- if $y_i \notin Z_3^* \rightarrow$ error = 0.5

same chance to have correct or wrong

wrong

$$\hat{Err}_{best} = 0 \Pr[Y_i \in Z_3^*] + 0.5 \Pr[Y_i \notin Z_3^*] =$$

$$\Pr[Y_i \notin Z_3^*] = \Pr[Y_i \neq Z_3^*[1]] \cdot \Pr[Y_i \neq Z_3^*[2]] \cdot \dots + \Pr[Y_i \neq Z_3^*[n]]$$

$$\left(\frac{n-1}{n}\right) \times \left(\frac{n-1}{n}\right) \times \dots \times \left(\frac{n-1}{n}\right)$$

$$\left(\frac{n-1}{n}\right)^n \approx e^{-1} \approx 0.368$$

$$= 0 + 0.5 \times 0.368 = 0.184 < 0.5$$

$$\Pr[Y_i \in Z_3^*] = 0.632$$

$P[Y_i \in Z_S^*] = 0.632$, that means that we only use 0.632 n unique observations to fit the model $\Rightarrow$ same overestimation problems of 2-CV

$$\text{err}_{0.632\text{boot}} = 0.368 \text{err}_{\text{train}} + 0.632 \text{err}_{\text{bootstrap}}$$

$\hat{\text{err}}_{(0.632)}$ in practice works really well
 $\rightarrow$ there are specific examples in which it fails

for example

$$\text{err}_{0.632\text{boot}} = 0.368 \cdot 0 + 0.632 \times 0.5 = 0.316 < 0.5$$

Solution: 0.632 + bootstrapping

$$\text{err}_{0.632+\text{boot}} = (1 - \hat{w}) \cdot \text{err}_{\text{train}} + \hat{w} \text{err}_{\text{bootstrap}}$$

$\hat{w}$ is a weight that controls for the amount of overfitting in the train data.
 $\rightarrow$ in our example, ~~the~~ our method fully overfit, so $\hat{w} = 1$

$$\text{err}_{0.632+\text{boot}} = (1 - 1) \text{err}_{\text{train}} + 1 \text{err}_{\text{train}} = 0.5 !$$

Methods based on the estimation of the optimism

- introduce the concept of optimism

$\text{err}_{\text{train}}$ = train error, underestimate the prediction error

$\overline{\text{err}}$ → optimism = difference between prediction error and the training error

- for a reasonable amount of loss function, specifically for the squared loss

$$\text{optimism} = \frac{2}{n} \sum_{i=1}^n \text{Cov}(y_i, \hat{f}(x_i)) \quad \frac{\frac{1}{n} \sum (y_i - \hat{f}(x_i))^2}{\hat{f}_i}$$

- optimism depends on how much y affects its own estimation
- the stronger the overfit, the larger $\text{Cov}(y_i, \hat{f}(x_i))$, the higher the optimism

When $\hat{f}(x)$ is obtained by a linear fit (e.g., linear regression)

$$\sum_{i=1}^n \text{Cov}(y_i, \hat{f}(x_i)) = p\sigma^2 \quad y = f(x) + \varepsilon, \text{Var}(\varepsilon) = \sigma^2$$

So

$$\begin{aligned} \text{error} &= \text{err}_{\text{train}} + \widehat{\text{optimism}} \\ &= \text{err}_{\text{train}} + \frac{2}{n} p\sigma^2 \end{aligned}$$

In the case of squared loss, $\underline{C_p} = \text{err}_{\text{train}} + \frac{2}{n} p\hat{\sigma}^2$

In the case of ~~the~~ linear Gaussian regression, $C_p = \text{AIC}$.

AIC is more general, and use ^{2nd known} _{negative} log-likelihood for the error

$$\text{AIC} = \underbrace{-\frac{2}{n} \log \text{lik}}_{\text{err}_{\text{best}}} + \underbrace{\frac{2}{n} p}_{\text{opt}}$$

When we want to find the ^{best} suitable value for a tuning parameter

$$\text{AIC}(\alpha) = -\frac{2}{n} \log \text{lik}(\alpha) + \frac{2}{n} p\hat{\sigma}^2$$

- this is the basic interpretation of AIC AIC actually comes from asymptotic computation based on the K-L divergence

$$AIC = \text{err}_{\text{train}} + \frac{2}{n} p$$

We can also see BIC from this perspective

$$BIC = \text{err}_{\text{train}} + \frac{\log n}{n} p \approx 2$$

AIC vs BIC

When $n > e^2 \approx 7.4$ (when $n \geq 8$), then $\log n > 2$
 $n > e^2$
 BIC penalizes more than AIC 'larger models (more complex)'
 → for $n \geq 8$, BIC leads to the selection of sparser models (simpler models)
 → often AIC is used for prediction, BIC for description
 predictive models descriptive models

Using AIC in regression, is approximately like testing at level $\alpha = 0.157$

$$\text{Likelihood Gaussian} = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2} (y_i - \hat{f}(x_i))^2\right\}$$

$$\text{suppose } \sigma^2 \text{ known} = \prod_{i=1}^n \exp\left\{-(y_i - \hat{f}(x_i))^2\right\}$$

$$\text{log-likelihood} = -\sum_{i=1}^n (y_i - \hat{f}(x_i))^2$$