Machine Learning

5. Evaluation

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Business Economics and Information Systems
& Institute for Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Error measures err (= loss functions L)

Numerical target y:

$$\text{err}(y, \hat{f}) = \begin{cases} (y - \hat{f}(x))^2 & \text{squared error} \\ |y - \hat{f}(x)| & \text{absolute error} \end{cases}$$

$$\text{err}(D, \hat{f}) = \begin{cases} \sqrt{\frac{1}{n}\sum_{i=1}^n (y_i - \hat{f}(x_i))^2} & \text{root mean squared error (RMSE)} \\ \frac{1}{n}\sum_{i=1}^n |y_i - \hat{f}(x_i)| & \text{mean absolute error (MAE)} \end{cases}$$

Nominal target y:

$$\text{err}(y, \hat{f}) = I(y = \hat{f}(x)) \qquad \text{0-1 loss}$$

$$\text{err}(D, \hat{f}) = \frac{1}{n}\sum_{i=1}^n I(y_i = \hat{f}(x_i)) \qquad \text{accuracy}$$

Both types of targets y:

$$\text{err}(y, \hat{p}(Y, x)) = -2 \log \hat{p}(y, x) \qquad \text{log likelihood / cross entropy / deviance}$$

$$\text{err}(D, \hat{p}) = -2\frac{1}{n}\sum_{i=1}^n \log \hat{p}(y_i, x_i) \qquad \text{log likelihood / cross entropy / deviance}$$

Training Error

Data D_{train} used to learn the model $\hat{f}$ is called **training data**.

The error on the training data is called **training error** (also **model fit**):

$$\text{fit}(\hat{f}) := \text{err}(D_{\text{train}}, \hat{f})$$

Most models are **universal approximators**, i.e., they can be configured s.t. the training error is zero (unless there are cases with the same x but different y):

- linear models: use as many derived variables as cases,
- nearest neighbor models: use 1-nearest neighbor.
- decision trees: allow pure leaves only.
- ...

Test Error

Therefore, more interesting is the error to be expected if the model is applied to fresh data:

$$\text{Err}(\hat{f}) := E_{X,Y}(\text{err}(Y, \hat{f}(X)))$$

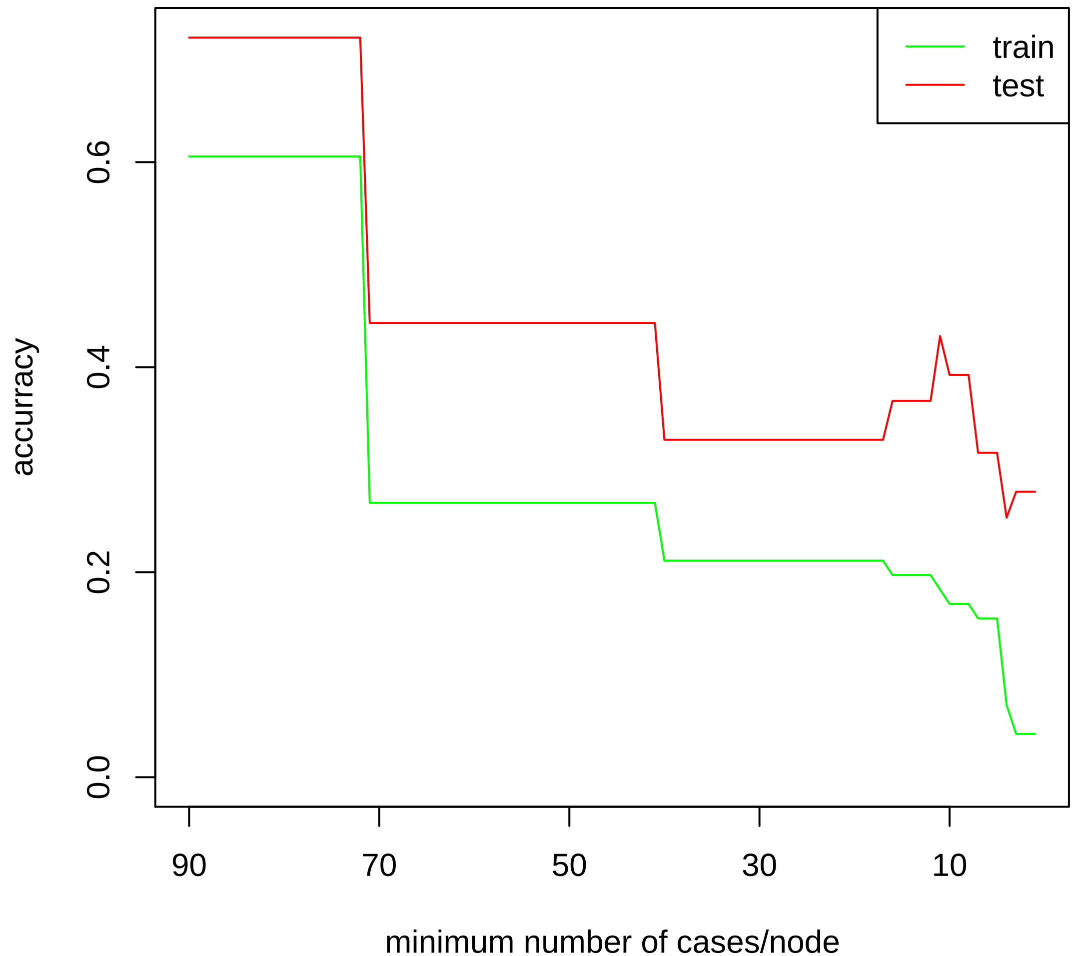
called **test error** (**generalization error**).

The test error is not accessible
(because the true distribution is not known).

But the test error can be estimated using fresh data D_{test}, called **test data**:

$$\widehat{\text{Err}}(\hat{f}) = \text{err}(D_{\text{test}}, \hat{f})$$

Train vs. Test Error / Example: Decision Tree on Iris Data (50/50 split)



Hyperparameters and Calibration Data Error

Whenever a learning process depends on a hyperparameter such as the minimum number of cases/node for decision trees, the hyperparameter can be estimated by picking the value with the lowest error.

If this is done on test data, one actually uses test data in the training process ("train on test"), thereby lessen its usefullness for estimating the test error.

Therefore, one splits the training data again in

- (proper) training data and
- **calibration data**.

The calibration data figures as test data during the training process.

Hyperparameters / Grid Search

Hyperparameters often are learnt simply by trying some values in a reasonable range and then pick the one with the lowest calibration error.

If there are more than one hyperparameter, say λ and μ, then all combinations are tried (**grid search**).

Hyperparameters usually learnt this way are

- the complexity parameter λ in ridge regression,
- the number of nearest neighbors k in nearest neighbor models,
- the kernel width λ in kernel regression,
- the minimum number of cases/node in decision trees (and eventually more regularization parameters such as the maximum depth etc.)
- etc.

Model Structure

Also model structures such as the predictor variables used in a model can be choosen by calibration error, e.g., backward and forward search in linear regression also can use calibration error instead of BIC.

1. Train and Test Errors

2. The Bias–Variance Decomposition

3. Cross Validation

4. The Bootstrap

The Bias–Variance Decomposition

Let

$$Y = f(X) + \epsilon, \quad E(\epsilon) = 0, V(\epsilon) = \sigma_\epsilon^2$$

and $\hat{f}$ be a model for f:

$$\begin{aligned}
\text{Err}(\hat{f}, x) &= E((Y - \hat{f}(x))^2 \,|\, X = x) \\
&= E((\epsilon + f(x) - \hat{f}(x))^2) \\
&= E(\epsilon^2) + E((f(x) - \hat{f}(x))^2) \\
&= \sigma_\epsilon^2 + (E\hat{f}(x) - f(x))^2 + E((\hat{f}(x) - E\hat{f}(x))^2) \\
&= \sigma_\epsilon^2 + \text{Bias}^2(\hat{f}(x)) + V(\hat{f}(x)) \\
&= \text{irreducible error} + \text{Bias}^2 + \text{Variance}
\end{aligned}$$

where

σ_ϵ^2 is the error due to variability in the true distribution – this cannot be reduced.

Bias is the error due to differences in the average true and estimated values,

$V(\hat{f}(x))$ is the error due to variability in the estimates.

The Bias–Variance Decomposition / k-nearest neighbor

$$\begin{aligned}\text{Err}(\hat{f}, x) &= \sigma_\epsilon^2 + (E\hat{f}(x) - f(x))^2 + E((\hat{f}(x) - E\hat{f}(x))^2) \\ &= \sigma_\epsilon^2 + (f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_{(i)}))^2 + \sigma_\epsilon^2/k\end{aligned}$$

Increase k:

$\rightsquigarrow$ increase bias
– the model can adapt less easily to f at a specific point x

$\rightsquigarrow$ decrease variance.

The Bias–Variance Decomposition / linear model

$$\hat{f}(x) = \langle \hat{\beta}, x \rangle, \quad \hat{\beta} \in \mathbb{R}^p$$

$$V(\hat{\beta}) = (X^T X)^{-1} \sigma_\epsilon^2$$

$$V(\hat{f}(x)) = ||x^T (X^T X)^{-\frac{1}{2}}||^2 \sigma_\epsilon^2 \text{ depends on } x\text{, but}$$

$$\frac{1}{n} \sum_{i=1}^n V(\hat{f}(x_i)) = \frac{p}{n} \sigma_\epsilon^2$$

and hence

$$\frac{1}{n} \sum_{i=1}^n \text{Err}(\hat{f}, x_i) = \sigma_\epsilon^2 + \frac{1}{n} \sum_{i=1}^n (E\hat{f}(x_i) - f(x_i))^2 + E((\hat{f}(x_i) - E\hat{f}(x_i))^2)$$

$$= \sigma_\epsilon^2 + \frac{1}{n} \sum_{i=1}^n (E\hat{f}(x_i) - f(x_i))^2 + \frac{p}{n} \sigma_\epsilon^2$$

Cross Validation

Instead of a single split into

training data, (validation data,) and test data

cross validation splits the data in k parts (of roughly equal size)

$$D = D_1 \cup D_2 \cup \cdots \cup D_k, \quad D_i \text{ pairwise disjunct}$$

and averages performance over k learning problems

$$D^{(i)}_{\text{train}} = D \setminus D_i, \quad D^{(i)}_{\text{test}} = D_i \quad i = 1, \ldots, k$$

Common is 5- and 10-fold cross validation.

n-fold cross validation is also known as **leave one out**.

Cross Validation

How many folds to use in k-fold cross validation?

$k = n$ / **leave one out:**

- approximately unbiased for the true prediction error.
- high variance as the n training sets are very similar.
- in general computational costly as n different models have to be learnt.

$k = 5$**:**

- lower variance.
- bias could be a problem,
  due to smaller training set size the prediction error could be overestimated.

...