Cross-validation[1]

Daniel Berrar

Machine Learning Research Group
School of Mathematics and Statistics
The Open University, Milton Keynes, United Kingdom
Email: daniel.berrar@open.ac.uk
and
Department of Information and Communications Engineering, School of Engineering,
Tokyo Institute of Technology, Tokyo, Japan

Abstract

Cross-validation is one of the most widely used data resampling methods for model selection and evaluation. Cross-validation can be used to tune the hyperparameters of statistical and machine learning models, to prevent overfitting, to compare learning algorithms, and to estimate the generalization error of predictive models. This article gives an introduction to the most common types of cross-validation, such as k-fold cross-validation, nested cross-validation, and leave-one-out cross-validation, as well as their relation to other data resampling strategies.

Keywords: classifier; classification; data resampling; flat cross-validation; fold; generalization error; inner fold; jackknife; k-fold cross-validation; k-fold random subsampling; learning set; leave-one-out cross-validation; LOOCV; nested cross-validation; outer fold; overfitting; prediction error; resubstitution error; single hold-out subsampling; test error; test set; training set; underfitting; validation set

Key points

- This article gives an introduction to cross-validation and related data resampling strategies for model selection and evaluation.
- The focus is on k-fold cross-validation and its variants, including stratified cross-validation, repeated cross-validation, nested cross-validation, and leave-one-out cross-validation.
- Some caveats and pitfalls of cross-validation are discussed.

[1] This article is the revised version of [6] for the 2nd edition of the *Encyclopedia of Bioinformatics and Computational Biology*, Elsevier.

1. Introduction

Cross-validation is one of the most widely used data resampling methods for model selection and evaluation. Cross-validation is used to assess the generalization ability of predictive models and to prevent overfitting [1, 2]. Like the bootstrap [3], cross-validation belongs to the family of Monte Carlo methods.

Consider a data set D, which consists of n p-dimensional instances (or cases), $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})$, $i = 1...n$. Each case is described by a set of p attributes (or features). We assume that each case belongs to one (and only one) class $y \in \{y_1, y_2, ..., y_c\}$; thus, the cases are labeled, $(\mathbf{x}_i, y_i)$. A typical example from bioinformatics is a gene expression data set based on DNA microarray data, where each case represents one labeled tumor sample described by a gene expression profile. One of the common challenges concerns the development of a classifier that can reliably predict the class of new, unseen tumor samples based on their expression profiles [4]. Conceptually, a classifier, f, is a rule for assigning a class label to a case based on a data set D, i.e., $f(\mathbf{x}_i, D) = \hat{y}_i$, where $\hat{y}_i$ is the predicted class label for case $\mathbf{x}_i$. In machine learning, the construction—or learning—of such a model is denoted as *supervised learning.*

A central question in supervised learning concerns the generalization ability of the resulting model. Here, a key problem is *overfitting* [5]. It is very easy to build a model that is perfectly adapted to the data set at hand but then unable to generalize well to new, unseen data. For example, consider a univariate regression problem where we wish to predict the dependent variable y from the independent variable x based on a n observations (x_i, y_i), with $i = 1...n$. We could use a polynomial of degree $n - 1$ to fit a curve perfectly through these points and then use the curve to extrapolate the value y_{i+1} for a new case, x_{i+1}. However, this curve is very likely to be overfitted to the data at hand—not only does it reflect the relation between the dependent and independent variable, but it has also modeled the inherent noise in the data set. On the other hand, a simpler model, such as a least squares line, is less affected by the inherent noise, but it may not capture well the relation between the variables, either. Such a model is said to be *underfitted.* Neither the overfitted nor the underfitted model are expected to generalize well, and a major challenge is to find the right balance between over- and underfitting.

How can we assess the generalization ability of a model? Ideally, we would evaluate the model using new data that originate from the same population as the data that we used to build the model [7]. In practice, new independent validation studies are often not feasible, though. Also, before we invest time and other resources for an external validation, it is advisable to estimate the predictive performance first. This is usually done by data resampling methods, such as cross-validation.

2. Basic concepts and notation

It is expedient to first clarify several key terms and concepts. Here, the term *model* refers to a statistical or machine learning model that results from the application of an algorithm to a data set; simply put, model = data + algorithm. A learning algorithm typically has *parameters* and *hyperparameters.* The parameters are internal to the algorithm, for example, the weights in a neural network or the coefficients in a linear regression model. During the training (or fitting) process, these parameters are adjusted automatically. By contrast, the hyperparameters control the training process. Examples of hyperparameters are the number of hidden layers in a multilayer perceptron, or the number k of neareast neighbors in a k-NN classifier. Hyperparameters are usually set manually and optimized by using data resampling strategies. *Model selection* consists of two parts: (1) selecting a learning algorithm for a concrete task and data set, and (2) tuning the hyperparameters of a learning algorithm. How can we find the optimal hyperparameters of a learning algorithm? How can we compare the performance of different learning algorithms for a concrete task? These questions can be addressed by using cross-validation or other types of random subsampling.

The data set that is available to build and evaluate a model is referred to as the *learning set*, L. This data set is assumed to be a random sample from a population of interest. Random subsampling methods are used to generate *training set(s)*, R, and *test set(s)*, T, from the learning set L. This sampling should be done in such a way that the training and test sets have no cases in common, i.e., $R \cap T = \emptyset$ (i.e., R and T are disjoint), and $L = R \cup T$. The model is then trained (or fitted) on the training set and tested on the test set(s). For example, consider Fisher's Iris data set, which consists of 150 instances of three species of Iris flowers (50 cases of Iris setosa, 50 cases of Iris virginica, and 50 cases of Iris versicolor). The entire data set of 150 cases represents the learning set L, which is assumed to be random sample from a population of Iris flowers. A possible test set could consists of 30 cases that were randomly selected from L, whereas the corresponding training set could consist of the remaining 120 cases.

The various random subsampling methods differ with respect to how the training and test sets are generated, and also with respect to how many such sets are generated. In cross-validation, the learning set is split into disjoint training sets, R_i, and validation sets, V_i. The term "validation set" is typically (but not always) used in the context of cross-validation, whereas the term "test set" commonly refers to a data set that is put aside for a final model evaluation (hold-out test set), that is, a data set that is used only for a final evaluation and that plays no role in the fitting process.

The term "training" implies that we apply the learning algorithm to a subset of the data, the training set R. The resulting model, $\hat{f}(\mathbf{x}, R)$, is only an estimate of the final model that results from applying the same learning algorithm to the entire learning set, $f(\mathbf{x}, L)$. Model evaluation based on repeated subsampling means that a learning algorithm is applied to several data subsets, and the resulting models, $\hat{f}_j$, are subsequently evaluated on other subsets (i.e., the test sets or validation sets), which were not used during training. The average of the performance that the models achieve on these subsets is an estimate of the performance of the final model, $f(\mathbf{x}, L)$.

Let us assume that with each case, exactly one class label y_i is associated. In the case of classification, y_i is a discrete class label. A classifier is a special case of a predictive model that assigns a discrete class label to a case. In regression tasks, the target is a real value, $y_i \in \mathbb{R}$. A predictive model f estimates the target y_i of the case $\mathbf{x}_i$ as $f(\mathbf{x}_i) = \hat{y}_i$. A loss function, $\mathcal{L}(y_i, \hat{y}_i)$, quantifies the estimation error. For example, using the 0-1 loss function in a classification task, the loss is 1 if $y_i \neq \hat{y}_i$ and 0 otherwise. There exists a variety of loss functions for binary classification; for an overview, see [8]. With the loss function, we can now calculate two different errors, (1) the *training error* and (2) the *test error.* The training error (or *resubstitution error*) tells us something about the adaptation to the training set(s), whereas the test error is an estimate of the true prediction error. This estimate quantifies the generalization ability of the model. Note that the training error tends to underestimate the true prediction error, since the same data that were used to train the model are reused to evaluate the model.

3. Feature selection in data resampling and avoiding information leak

Feature selection is generally an integral part of the model building process. Here, it is crucial that predictive features are selected using only the training set, not the entire learning set; otherwise, the estimate of the prediction error can be highly biased [9, 10]. Suppose that predictive features are selected based on the entire learning set first, and then the learning set is partitioned into test sets and training sets. This means that information from the test sets was used for the selection of predictive features. But the data in the test sets serve only to evaluate the model—we are not allowed to use these data in any other way; otherwise, the information leak would cause a downward bias of the estimate, which means that it underestimates the true prediction error. To avoid a possible information leak, feature selection should therefore be performed only on the training data, not on data that are used to evaluate the model.

4. Data resampling strategies

4.1. Single hold-out random subsampling

Among the various data resampling strategies, one of the simplest ones is the *single hold-out method*, which randomly samples some cases from the learning set for the test set, while the remaining cases constitute the training set. Often, the test set contains about 10% to 30% of the available cases, and the training set contains about 90% to 70% of the remaining cases, respectively. If the learning set is sufficiently large, and consequently, if both the training and test sets are relatively large, then the observed test error can be a reliable estimate of the true error of the model for new, unseen cases.

4.2. k-fold random subsampling

In *k-fold random subsampling*, the single hold-out method is repeated k times, so that k pairs of training sets R_j and test sets T_j, $j = 1...k$, are generated. The learning algorithm is applied to each training set, and the resulting model is then applied to the corresponding test set. The performance is estimated as the average over all k test sets. Note that any pair of training and test set is disjoint, i.e., the sets do not have any cases in common. However, any given two training sets or two test sets may of course overlap.

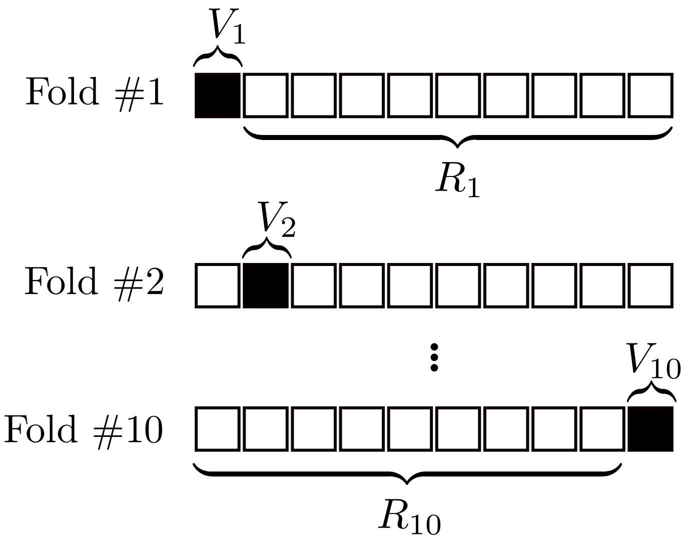


Figure 1: k-fold cross-validation, with $k = 10$. The data set is randomly split into 10 disjoint subsets, each containing 10% of the data. The classifier is trained on the training set R_i and then applied to the validation set V_i.

4.3. k-fold cross-validation

k-fold cross-validation is similar to the repeated random subsampling method, but the sampling is done in such a way that no two test sets overlap. In k-fold cross-validation, the available learning set is partitioned into k disjoint subsets of approximately equal size. Here, "fold" refers to the number of resulting subsets. This partitioning is performed by randomly sampling cases from the learning set without replacement. The model is trained on $k-1$ subsets, which, together, represent the training set. Then, the model is applied to the remaining subset, which is denoted as the *validation set*, and the performance is measured. This procedure is repeated until each of the k subsets has served as validation set. The average of the k performance measurements on the k validation sets is the cross-validated performance. Figure 1 illustrates this process for $k = 10$, i.e., 10-fold cross-validation. In the first fold, the first subset serves as validation set V_1 and the remaining nine subsets serve as training set R_1. In the second fold, the second subset is the validation set and the remaining subsets are the training set, and so on.

The cross-validated accuracy, for example, is the average of all ten accuracies achieved on the validation sets. More generally, let $\hat{f}_{-j}$ denote the model that was trained on all but the j^{th} subset of the learning set, with $j = 1...k$. The value $\hat{y}_i = \hat{f}_{-j}(\mathbf{x}_i)$ is the predicted or estimated value for the real class label, y_i, of case $\mathbf{x}_i$, which is an element of the j^{th} subset. The cross-validated estimate of the prediction error, $\hat{\epsilon}_{\text{cv}}$, is then given as

$$\hat{\epsilon}_{\text{cv}} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, \hat{f}_{-j}(\mathbf{x}_i)) \tag{1}$$

where n is the number of cases, and $\mathcal{L}$ is a loss function.

Cross-validation often involves *stratified random sampling*, which means that the sampling is performed in such a way that the class proportions in the individual subsets reflect the proportions in the learning set. For example, suppose that the learning set contains $n = 100$ cases of two classes, the positive and the negative class, with $n_+ = 80$ and $n_- = 20$. If random sampling is done without stratification, then it is possible that some validation sets contain only positive cases. With stratification, however, each validation set in 10-fold cross-validation is guaranteed to contain about eight positive cases and two negative cases, thereby reflecting the class ratio in the learning set. The underlying rationale for stratified sampling is the following. The sample proportion is an unbiased estimate of the population proportion. The learning set represents a sample from the population of interest, so the class ratio in the learning set is the best estimate for the class ratio in the population. To avoid a biased evaluation, data subsets that are used for evaluating the model should therefore also reflect this class ratio. For real-world data sets, stratified 10-fold cross-validation is recommended [11].

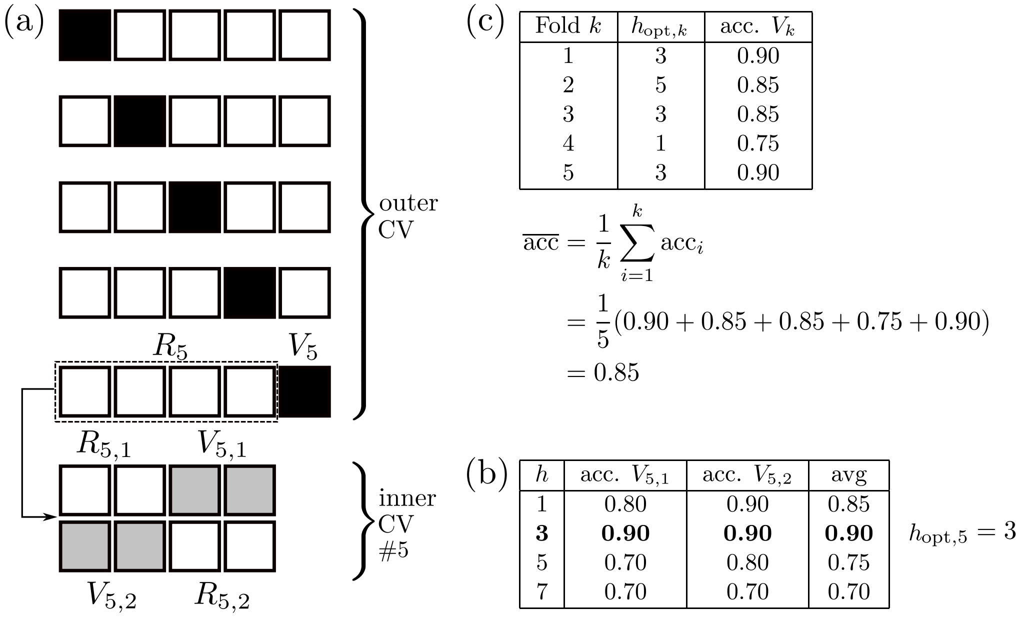
To reduce the variance of the estimated performance measure, cross-validation is sometimes repeated with different k-fold subsets, which is referred to as *r times repeated k-fold cross-validation*. However, Molinaro et al. showed that such repetitions reduce the variance only slightly [12].

4.4. Nested cross-validation

Nested cross-validation is a special case of ordinary k-fold cross-validation (CV) that involves one outer CV and several inner CVs [13]. The outer CV is the same as the k-fold cross-validation described in Section 4.3. Each fold of the outer CV involves an internal cross-validation that uses the respective training set. Thus, there is an additional, nested cross-validation within each fold of the ordinary cross-validation. This nesting is schematically described in Figure 2a, which shows a commonly used 5×2 nested cross-validation. In this example, we assume that the performance measure is accuracy, but in principle any performance measure could be used.

In Figure 2a, the outer CV consists of five folds, and each of these folds consists of one inner CV with two folds. For simplicity, only the inner CV of the fifth outer fold is shown. Here, the outer training set and outer validation set are denoted by R_5 and V_5, respectively. In the inner CV, a two-fold cross-validation is performed on R_5. In the first fold of the inner CV, R_5 is split into the inner training set $R_{5,1}$ and inner validation set $V_{5,1}$. In the second fold, these sets are swapped: $R_{5,1}$ becomes $V_{5,2}$, and $V_{5,1}$ becomes $R_{5,2}$ (Figure 2a). The purpose of the inner CV is to find the best hyperparameters of a learning algorithm. For simplicity, let us assume that a learning algorithm has only one tunable hyperparameter, h, for example, the number of k neareast neighbor of a k-NN algorithm [14]. Let us further assume that we consider only four values, $h \in \{1, 3, 5, 7\}$.[2] In each inner CV of 5×2 cross-validation, we obtain two performance measurements per hyperparameter value. In Figure 2b, we obtain an accuracy of 0.80 on $V_{5,1}$ and 0.90 on $V_{5,2}$ for $h = 1$,

[2] We denote the number of nearest neighbors by h and not by k to avoid confusion with k-fold, the number of outer folds.



Fold k	$h_{\text{opt},k}$	acc. V_k
1	3	0.90
2	5	0.85
3	3	0.85
4	1	0.75
5	3	0.90

$$\begin{aligned}\overline{\text{acc}} &= \frac{1}{k}\sum_{i=1}^{k} \text{acc}_i \\ &= \frac{1}{5}(0.90 + 0.85 + 0.85 + 0.75 + 0.90) \\ &= 0.85\end{aligned}$$

h	acc. $V_{5,1}$	acc. $V_{5,2}$	avg
1	0.80	0.90	0.85
3	**0.90**	**0.90**	**0.90**
5	0.70	0.80	0.75
7	0.70	0.70	0.70

$h_{\text{opt},5} = 3$

Figure 2: (a) 5×2 nested cross-validation (CV). The outer CV is a 5-fold cross-validation to evaluate the model fitting procedure. Each row represents the available data sets; dark blocks represent subsamples used for validation, while white blocks represent subsamples used for training. Each fold consists of an inner CV to find the optimal hyperparameter, h. The inner CV is only shown for the fifth fold, where R_5 indicates the training set and V_5 indicates the validation set. The fifth training set is split into an inner training set, $R_{5,1}$ and an inner validation set $V_{5,1}$, which are then reversed. (b) For the hyperparameter h, four values are used in the fifth inner CV, $h \in \{1, 3, 5, 7\}$. The hyperparameter that results in the highest average accuracy in this example is $h_{\text{opt},5} = 3$. All accuracies are contrived numbers and serve for numeric illustration only. (c) Optimal hyperparameters $h_{\text{opt},k}$ for all $k = 5$ folds of the outer CV. See main text for explanation.

with a mean of 0.85. For $h = 3$, the mean accuracy is 0.90, and so on. The optimal value for the fifth inner CV is $h_{\text{opt},5} = 3$, since the average accuracy is the highest.

The optimal hyperparameter $h_{\text{opt},k}$ is then used for the model that is to be trained on R_k. This model is then applied to V_k. In the example shown in Figure 2c, $h_{\text{opt},5} = 3$ leads to an accuracy of 0.90 on V_5. By proceeding analogously for all five outer folds, we achieve a cross-validated accuracy of 0.85.

The optimal values of the hyperparameters are not necessarily the same for each outer fold, as shown in Figure 2b. The cross-validated accuracy of 0.85 is therefore an average of *different, but similar*, models, since these models were built using different hyperparameters. Also, note that despite the substantial overlap between the training sets R_1, R_2, R_3, R_4, and R_5, the models were built on different data subsets. The variance of the performance values in the different validation sets informs us about the stability of the fitted model; a low variance points to a high model stability, whereas a high variance points to a low model stability.

The observed cross-validated performance is an estimate of the performance that results from the application of the learning algorithm to this data set. For example, let us assume again that the learning algorithm used in this example is k-NN. The accuracy of the k-NN algorithm on this learning set, L, is therefore estimated as $\widehat{\text{acc}}(k\text{-NN algorithm}, L) = 0.85$. Suppose that, in addition to k-NN, we also included another learning algorithm, for example, a support vector machine (SVM), for which we observe a cross-validated accuracy of $\widehat{\text{acc}}(\text{SVM algorithm}, L) = 0.72$. Taking into account only the performance, we might decide that the k-NN algorithm is more suitable to this data set, since it achieved a higher accuracy. Thus, we have used nested cross-validation with a dual purpose, (1) tuning the hyperparameters, and (2) selecting the most suitable learning algorithm for a concrete classification task. Note that the algorithms should be applied to the exact same data subsamples, that is, in the same nested cross-validation. If they are applied in different executions of the nested cross-validation, then they are effectively trained on different random subsamples, which would not represent a fair comparison.

After the selection of the "winner" algorithm, we still do not have a final model that can be deployed for new, unseen data. In fact, which value of h should be chosen to train a k-NN model on the entire learning set? The choice would be easy if all five $h_{\text{opt},k}$ were the same, as this value could then be used. In practice, the optimal hyperparameter (or set of hyperparameters) can be different from fold to fold, though. It would not be advisable to select that value for which the best performance among the five outer folds was achieved because this approach would ignore the results from the remaining four folds and could be just due to chance. Instead, to find the "best" hyperparameter for the "winner" algorithm, it is preferable to run additional ordinary k-fold cross-validations (that is, without nesting), one for each distinct value of $h_{\text{opt},k}$. The hyperparameter for which the best cross-validated performance is achieved will then be selected. For example, in Figure 2c, the values $h \in \{1, 3, 5\}$ could be checked in 10-fold cross-validation. Suppose that the highest cross-validated accuracy of 0.88 is achieved for $h = 3$. Then this value will be used for the hyperparameter when the model is trained on the entire learning set, L. Let D_{new} denote a new, randomly sampled data set from the same population as that from which L originates. The performance of the model on D_{new} is therefore estimated as $\widehat{\text{acc}}(\text{3-NN}, D_{\text{new}}) = 0.88$.

A natural question is why the best algorithm and hyperparameters could not be found directly in ordinary cross-validation without nesting. This procedure is referred to as *flat cross-validation* [15]. For example, suppose that we have the choice between a k-NN algorithm (with hyperparameter $h \in \{1, 3, 5, 7\}$) and a support vector machine (with hyperparameter kernel $\in \{\text{linear}, \text{polynomial}\}$). In 10-fold cross-validation, we could evaluate 1-NN, 3-NN, 5-NN, 7-NN, $\text{SVM}_{\text{linear}}$, and SVM_{poly}; the model with best cross-validated performance is then declared the "winner". This approach is computationally less expensive than nested cross-validation, but it introduces an optimistic bias into the performance estimate, since the validation sets were used for *both* hyperparameter tuning *and* the selection of the learning algorithm [16]. Also, the comparison between k-NN and SVM is not really fair, as we considered four hyperparameter values for k-NN but only two for SVM. Despite these caveats, Wainer and Cawley showed that, in practice, a single flat cross-validation is often sufficient both for model selection (i.e., selecting the best learning algorithm and tuning the hyperparameters) and for estimating the expected performance [15].

The cross-validated accuracy is only a point estimate of the expected accuracy of the final model, and it can be useful to derive an interval for plausible values. However, deriving a confidence interval for a cross-validated measure is not trivial, as there exists no universal unbiased estimator of the variance of k-fold cross-validation [17]. The training sets in the different folds overlap, which means that the performance values that are observed on the validation sets are not independent. Jiang et al. developed a bootstrap case cross-validation resampling method for genomic microarray data classification [18]. A new method based on a nested cross-validation was recently proposed to estimate the variance, which can be used to calculate confidence intervals for cross-validated measures relatively accurately [19].

4.5. Statistical comparison of classifiers in cross-validation

A comparatively simpler method for addressing the variance problem was proposed for the comparison of two different classifiers in cross-validation: the *variance-corrected t-test* [20, 21]. In r times repeated k-fold cross-validation, the corrected t-statistic is

$$T = \frac{\frac{1}{kr}\sum_{i=1}^{k}\sum_{j=1}^{r}(a_{ij} - b_{ij})}{\sqrt{(\frac{1}{kr} + \frac{n_2}{n_1})s^2}} \sim t_{kr-1} \tag{2}$$

where a_{ij} and b_{ij} denote the performances achieved by two competing classifiers, A and B, respectively, in the j^{th} repetition of the i^{th} cross-validation fold; s^2 is the sample variance; n_2 is the number of cases in one validation set, and n_1 is the number of cases in the corresponding training set. This statistic follows approximately Student's t distribution with $k \times r - 1$ degrees of freedom. The variance-corrected t-test should be used carefully, though—by increasing r, even a tiny difference in performance can be made significant, which is misleading because essentially the same data are analyzed over and over again [22]. As the training sets in the different cross-validation folds overlap, the independence assumption of the standard t-test is

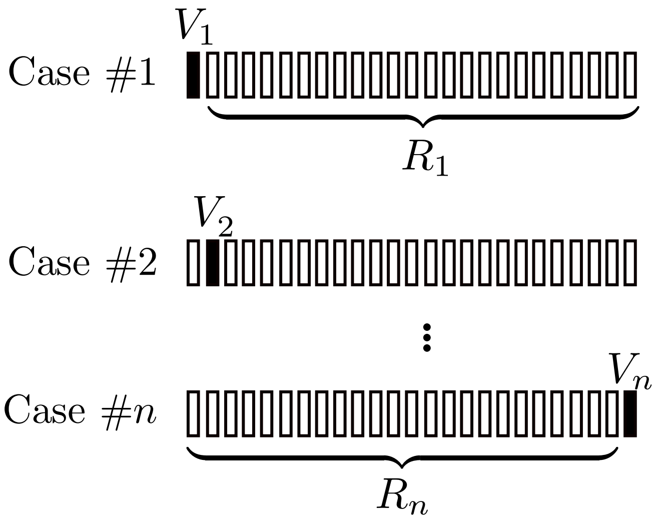


Figure 3: Leave-one-out cross-validation, illustrated on a data set containing $n = 25$ cases. In turn, each case serves as single hold-out test case. Each validation set therefore contains only one element: the first validation set is $V_1 = \{\mathbf{x}_1\}$, the second validation set is $V_2 = \{\mathbf{x}_2\}$, and so on; the last validation set is $V_n = \{\mathbf{x}_n\}$. The model is trained on the remaining $n - 1$ cases; hence, $R_i = L \setminus V_i$, $i = 1...n$.

violated, which leads to an underestimation of the variance; therefore, the standard t-test is not suitable for cross-validation.

4.6. Leave-one-out cross-validation

For $k = n$, we obtain a special case of k-fold cross-validation, called *leave-one-out cross-validation* (LOOCV). Here, each individual case serves, in turn, as hold-out case for the validation set. Thus, the first validation set contains only the first case, $\mathbf{x}_1$, the second validation set contains only the second case, $\mathbf{x}_2$, and so on. This procedure is illustrated in Figure 3 for a data set consisting of $n = 25$ cases. The test error in LOOCV is approximately an unbiased estimate of the true prediction error, but it has a high variance, since the n training sets are practically the same, as two different training sets differ only with respect to one case [1]. The computational cost of LOOCV can also be very high for large n, particularly if feature selection has to be performed.

4.7. Jackknife

Leave-one-out cross-validation is very similar to a related method, called the *jackknife* [23, 24, 25]. Essentially, these two methods differ with respect to their goal. Leave-one-out cross-validation is used to estimate the generalization ability of a predictive model. By contrast, the jackknife is used to estimate the bias or variance of a statistic, $\hat{\theta}$ [25]. Note that the available data set is only a sample from the population of interest, so $\hat{\theta}$ is only an estimate of the true parameter, θ. Another difference is that in LOOCV, the statistic of interest is calculated based on the test cases, whereas in the jackknife, the statistic is calculated based on the training cases.

The jackknife involves the following steps [2].

1. Calculate the sample statistic $\hat{\theta}$ as a function of the available cases $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, i.e., $\hat{\theta} = t(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, where $t()$ is a statistical function.
2. For all $i = 1...n$, omit the i^{th} case and apply the same statistical function $t()$ to the remaining $n-1$ cases and obtain $\hat{\theta}_i = t(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, ..., \mathbf{x}_n)$. Note that the index i means that the i^{th} case is *not* being used.
3. The jackknife estimate of the statistic $\hat{\theta}$ is the average of all $\hat{\theta}_i$, i.e., $\bar{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta}_i$.

The jackknife estimates of bias and variance of $\hat{\theta}$ are

$$\mathrm{bias}(\hat{\theta}) = (n-1)\left(\bar{\hat{\theta}} - \hat{\theta}\right) \tag{3}$$

$$\mathrm{Var}(\hat{\theta}) = \frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_i - \bar{\hat{\theta}}\right)^2 \tag{4}$$

5. Discussion

Cross-validation is a widely used data resampling method for model selection and evaluation. Cross-validation is used for tuning model hyperparameters, comparing different learning algorithms, and evaluating the performance of models. To fit the final model for the prediction or classification of real future cases, the learning algorithm is applied to the entire learning set; this final model cannot be cross-validated, though. Cross-validation provides an estimate for the performance of the final model on new data. Nested cross-validation disentangles hyperparameter tuning from the model evaluation, but in practice, the less computationally expensive ordinary (or flat) cross-validation is often sufficient despite its optimistic bias.

Which data resampling method should be used in practice? Molinaro et al. compared various data resampling methods for high-dimensional data sets, which are common in bioinformatics [12]. Their findings suggest that LOOCV, 10-fold cross-validation, and the .632+ bootstrap have the smallest bias. It is not clear, however, which value of k should be chosen for k-fold cross-validation. A sensible choice is probably $k = 10$, as the estimate of prediction error is almost unbiased in 10-fold cross-validation [10]. Isaksson et al., however, caution that cross-validated performance measures are unreliable for small-sample data sets and recommend that the true classification error be estimated using Bayesian intervals and a single hold-out test set [26]. Practitioners should keep in mind that data resampling is no panacea for fully independent validation studies involving independent test sets.

References

[1] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd edition, Springer, New York/Berlin/Heidelberg, 2008.

[2] R. Duda, P. Hart, D. Stork, Pattern Classification, 2nd Edition, John Wiley & Sons, USA, 2001.

[3] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, 1993.

[4] D. Berrar, M. Granzow, W. Dubitzky, Introduction to genomic and proteomic data analysis, in: W. Dubitzky, M. Granzow, D. Berrar (Eds.), Fundamentals of Data Mining in Genomics and Proteomics, Springer, 2007, pp. 1–37.

[5] D. Berrar, W. Dubitzky, Overfitting, in: W. Dubitzky, O. Wolkenhauer, K.-H. Cho, H. Yokota (Eds.), Encyclopedia of Systems Biology, Springer, 2013, pp. 1617–1619.

[6] D. Berrar, Cross-validation, in: S. Ranganathan, K. Nakai, C. Schönbach, M. Gribskov (Eds.), Encyclopedia of Bioinformatics and Computational Biology, 1st edition, Elsevier, 2018, pp. 542–545.

[7] R. Simon, Supervised analysis when the number of candidate features (p) greatly exceeds the number of cases (n), ACM SIGKDD Expl Newsletter 5 (2) (2003) 31–36.

[8] D. Berrar, Performance measures for binary classification, in: S. Ranganathan, K. Nakai, C. Schönbach, M. Gribskov (Eds.), Encyclopedia of Bioinformatics and Computational Biology, 1st edition, Elsevier, 2018, pp. 546–560.

[9] C. Ambroise, G. J. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, Proceedings of the National Academy of Sciences 99 (10) (2002) 6562–6566.

[10] R. Simon, Resampling strategies for model assessment and selection, in: W. Dubitzky, M. Granzow, D. Berrar (Eds.), Fundamentals of Data Mining in Genomics and Proteomics, Springer, 2007, pp. 173–186.

[11] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, Proceedings of the 14th International Joint Conference on AI (IJCAI), 1995, pp. 1137–1143.

[12] A. Molinaro, R. Simon, R. Pfeiffer, Prediction error estimation: a comparison of resampling methods, Bioinformatics 21 (15) (2005) 3301–3307.

[13] S. Varma, R. Simon, Bias in error estimation when using cross-validation for model selection, BMC Bioinformatics 7 (91) (2006).

[14] D. Berrar, I. Bradbury, W. Dubitzky, Instance-based concept learning from multiclass DNA microarray data, BMC Bioinformatics 7 (73) (2006).

[15] J. Wainer, G. Cawley, Nested cross-validation when selecting classifiers is overzealous for most practical applications, Expert Systems with Applications 182 (2021) 115222.

[16] G. Cawley, N. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, Journal of Machine Learning Research 11 (2010) 2079–2107.

[17] Y. Bengio, Y. Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, Journal of Machine Learning Research 5 (2004) 1089–1105.

[18] W. Jiang, S. Varma, R. Simon, Calculating confidence intervals for prediction error in microarray classification using resampling, Statistical Applications in Genetics and Molecular Biology 7 (1) (2008) Article8.

[19] S. Bates, T. Hastie, R. Tibshirani, Cross-validation: What does it estimate and how well does it do it?, Journal of the American Statistical Association (2023) 1–12doi:https://doi.org/10.1080/01621459.2023.2197686.

[20] C. Nadeau, Y. Bengio, Inference for the generalization error, Machine Learning 52 (2003) 239–281.

[21] R. Bouckaert, E. Frank, Evaluating the replicability of significance tests for comparing learning algorithms, Advances in Knowledge Discovery and Data Mining 3056 (2004) 3–12.

[22] D. Berrar, *Confidence curves*: an alternative to null hypothesis significance testing for the comparison of classifiers, Machine Learning 106 (6) (2017) 911–949.

[23] J. Tukey, Bias and confidence in not-quite large samples (abstract), The Annals of Mathematical Statistics 29 (2) (1958) 614.

[24] R. Miller, The jackknife — a review, Biometrika 61 (1) (1974) 1–15.

[25] B. Efron, C. Stein, The jackknife estimate of variance, The Annals of Statistics 9 (3) (1981) 586–596.

[26] A. Isaksson, M. Wallmana, H. Göransson, M. Gustafsson, Cross-validation and bootstrapping are unreliable in small sample classification, Pattern Recognition Letters 29 (14) (2008) 1960–1965.