Cross-validation

Daniel Berrar

Data Science Laboratory, Tokyo Institute of Technology
2-12-1-S3-70 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
Email: daniel.berrar@ict.e.titech.ac.jp

Abstract

Cross-validation is one of the most widely used data resampling methods to estimate the true prediction error of models and to tune model parameters. This article provides an introduction to the most common types of cross-validation and their related data resampling methods.

Keywords: data resampling; jackknife; k-fold cross-validation; k-fold random subsampling; learning set; leave-one-out cross-validation; overfitting; prediction error; resubstitution error; single hold-out subsampling; test error; training set; underfitting; validation set

1. Introduction

Cross-validation is a data resampling method 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. This article provides an introduction to cross-validation and its related resampling methods.

Consider a data set D, which consists of n labeled instances (or cases), x_i, $i = 1...n$. Each case is described by a set of attributes (or features). We assume that each case x_i belongs to exactly one class 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 predictive model, $f()$, is a rule for assigning a class label to a case based on a data set D, i.e., $f(x, D) = \hat{y}_i$, where $\hat{y}_i$ is the predicted class label for case x_i. In machine learning, the construction of such a model is denoted as *supervised learning*.

A central question in supervised learning concerns the accuracy 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 manuscript is the preprint of: Berrar D. (2018) Cross-validation. *Encyclopedia of Bioinformatics and Computational Biology*, Volume 1, Elsevier, pp. 542–545. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

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 [6]. 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. This article describes the major subtypes of cross-validation and their related resampling methods.

2. Basic concepts and notation

The data set that is available to build and evaluate a predictive model is referred to as the learning set, D_{learn}. This data set is assumed to be a sample from a population of interest. Random subsampling methods are used to generate training set(s), D_{train}, and test set(s), D_{test}, from the learning set. The model is then built (or trained) using the training set(s) and tested on the test set(s). The various random subsampling methods differ with respect to how the training and test sets are generated.

Note that the term "training" implies that we apply the learning algorithm to a subset of the data. The resulting model, $\hat{f}(x, D_{\text{train}})$, is only an estimate of the final model that results from applying the same learning function to the entire learning set, $f(x, D_{\text{learn}})$. Model evaluation based on repeated subsampling means that a learning function 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, $f(x, D_{\text{learn}})$.

Let us assume that with each case, exactly one target 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 usually a real value, $y_i \in \mathbb{R}$. A predictive model $f()$ estimates the target y_i of the case x_i as $f(x_i) = \hat{y}_i$. 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. 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.

2.1. 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 cases. If the learning set is sufficiently large, and consequently, if both the training and test sets are large, then the observed test error can be a reliable estimate of the true error of the model for new, unseen cases.

2.2. k-fold random subsampling

In k-fold random subsampling, the single hold-out method is repeated k times, so that k pairs of $D_{\text{train},j}$ and $D_{\text{test},j}$, $j = 1..k$, are generated. The learning function 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, $D_{\text{train},j} \cap D_{\text{test},j} = \emptyset$. However, any given two training sets or two test sets may of course overlap.

2.3. k-fold cross-validation

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 using $k-1$ 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 $D_{\text{val},1}$ and the remaining nine subsets serve as training set $D_{\text{train},1}$. 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 $\hat{f}_{-k}$ denote the model that was trained on all but the k^{th} subset of the learning set. The value $\hat{y}_i = \hat{f}_{-k}(x_i)$ is the predicted or estimated value for the real class label, y_i, of case x_i, which is an element of the k^{th} subset. The cross-validated estimate of the prediction error, $\hat{\epsilon}_{\text{cv}}$, is then given as

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

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.

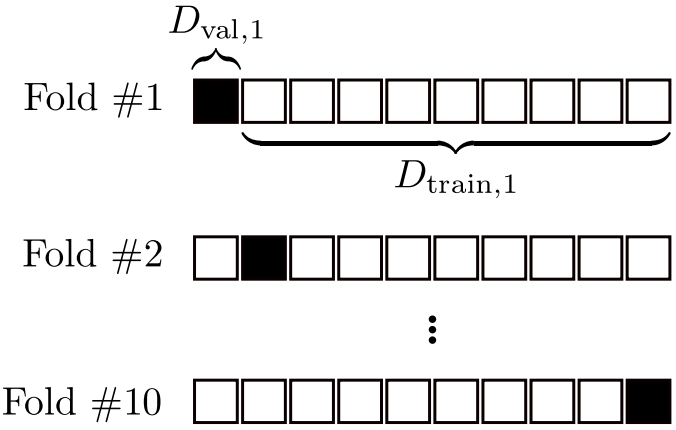

Figure 1: 10-fold cross-validation. The data set is randomly split into ten disjoint subsets, each containing (approximately) 10% of the data. The model is trained on the training set and then applied to the validation set.

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 quite possible that some validation sets contain only positive cases (or only negative 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, Kohavi recommends stratified 10-fold cross-validation [7].

To reduce the variance of the estimated performance measure, cross-validation is sometimes repeated with different k-fold subsets (r times repeated k-fold cross-validation). However, Molinaro et al. showed that such repetitions reduce the variance only slightly [8].

For the comparison of two different classifiers in cross-validation, the variance-corrected t-test was proposed [9, 10]. Note that the training sets in the different cross-validation folds overlap, which violates the independence assumption of the standard t-test and leads to an underestimation of the variance. The variance-corrected t-statistic is

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

This statistic follows approximately Student's t distribution with $k \cdot r - 1$ degrees of freedom. Here, 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 variance; n_2 is the number of cases in one validation set,

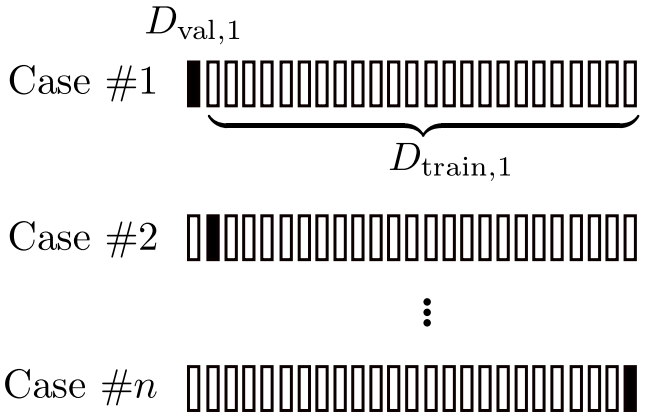

Figure 2: Leave-one-out cross-validation, illustrated on a data set containing $n = 25$ cases. In turn, each case serves as single hold-out test case. The model is built using the remaining $n - 1$ cases.

and n_1 is the number of cases in the corresponding training set. This 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 [11].

2.4. 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, x_1, the second validation set contains only the second case, x_2, and so on. This procedure is illustrated in Figure 2 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.

2.5. Jackknife

Leave-one-out cross-validation is very similar to a related method, called the *jackknife*. 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, $\hat{\theta}$ [12]. 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. The jackknife involves the following steps [2].

1. Calculate the sample statistic $\hat{\theta}$ as a function of the available cases x_1, x_2,..., x_n, i.e., $\hat{\theta} = t(x_1, x_2, ...x_n)$, 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 $\hat{\theta}_i = t(x_1, x_2, x_{i-1}, x_{i+1}, ...x_n)$. (NB: the index i means that the i^{th} case is *not* 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

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

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

3. Discussion

Cross-validation is one of the most widely used data resampling methods to assess the generalization ability of a predictive model and to prevent overfitting. To build the final model for the prediction of real future cases, the learning function (or learning algorithm) f is usually applied to the entire learning set. This final model cannot be cross-validated. The purpose of cross-validation in the model building phase is to provide an estimate for the performance of this final model on new data.

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 [13, 14]. Suppose that predictive features are selected based on the entire learning set first, and then the learning set is partitioned into validation sets and training sets. This means that information from the validation sets was used for the selection of predictive features. But the data in the validation 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.

Cross-validation is frequently used to tune model parameters, for example, the optimal number of nearest neighbors in a k-nearest neighbor classifier. Here, cross-validation is applied multiple times for different values of the tuning parameter, and the parameter that minimizes the cross-validated error is then used to build the final model. Thereby, cross-validation addresses the problem of overfitting.

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 [8]. 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 [14]. 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 [15].

4. Closing remarks

Cross-validation is one of the most widely used data resampling methods to estimate the true prediction error of models and to tune model parameters. Ten-fold stratified cross-validation is often applied in practice. Practitioners should keep in mind, however, that data resampling is no panacea for fully independent validation studies involving new data.

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, John Wiley & Sons, 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] 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.

[7] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143.

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

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

[10] 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.

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

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

[13] 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.

[14] 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.

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