Chapter 4

Cross-Validation

4.1 Introduction

The generalization performance (which is a terminology from the machine learning community) of a learning method describes its prediction capacity on new test data. Or in other words, the generalization performance measures the predictive power of a learning method on new, out-sample data. Having an estimate of this predictive power is very important for comparing among different methods or for tuning an estimator or algorithm to optimize predictive power.

Cross-Validation is one very general method for estimating such generalization or out-sample performance.

4.2 Training and Test Set

Consider the situation where we have data

$$(X_1, Y_1), \ldots, (X_n, Y_n) \text{ i.i.d. } \sim P$$

where P denotes the unknown distribution. (In case where the x_i is non-random, we can also drop the i.i.d. assumption for the pairs (x_i, Y_i).)

We typically have a target in mind, for example the regression function $m(\cdot)$. The estimated regression function $\hat{m}(\cdot)$ is then constructed by using some estimator or algorithm, based on the data $(X_1, Y_1), \ldots, (X_n, Y_n)$. This data is also called the **training data**, since it is used to train the estimator or learning algorithm.

We then would like to evaluate the accuracy of the estimated target which is based on the training data. A principal problem thereby is that if we use the training data again to measure the predictive power of our estimated target (e.g. of $\hat{m}(\cdot)$), the results will be overly optimistic. For example, when using the residual sum or squares in regression,

$$\sum_{i=1}^{n} (Y_i - \hat{m}(X_i))^2$$

becomes smaller the more "complex" (more degrees of freedom) the model for $\hat{m}(\cdot)$ involves. Obviously, a very complex model will not be good. We have seen already in Section 3.4 that penalizing the residual sum of squares can be useful to cope with the overfitting problem.

Alternatively, we could look how well the estimated target, e.g. $\hat{m}(\cdot)$, does on **new test data**

$$(X_{new,1}, Y_{new,1}), \ldots, (X_{new,\ell}, Y_{new,\ell}) \text{ i.i.d. } \sim P,$$

which is assumed to be independent from the training data but having the same distribution P. In case of regression, we would like to evaluate

$$\ell^{-1} \sum_{i=1}^{\ell} (Y_{new,i} - \hat{m}(X_{new,i}))^2.$$

More generally, when having an estimated target $\hat{m}$ and a loss function ρ, we would like to evaluate

$$\ell^{-1} \sum_{i=1}^{\ell} \rho\left(Y_{new,i}, \hat{m}(X_{new,i})\right),$$

where $\hat{m}(\cdot)$ is constructed from training data only. If ℓ is large, this evaluation on the test set approximates the theoretical test set error

$$\mathbb{E}_{(X_{new},Y_{new})}[\rho(Y_{new}, \underbrace{\hat{m}}_{\text{based on training data only}}(X_{new}))]$$

which is still a function of the training data. The expected value of this (with respect to the training data) is the generalization error,

$$\mathbb{E}_{training}\mathbb{E}_{(X_{new},Y_{new})}[\rho(Y_{new}, \underbrace{\hat{m}}_{\text{...training data}}(X_{new}))] = \mathbb{E}[\rho(Y_{new}, \hat{m}(X_{new}))], \tag{4.1}$$

where the latter $\mathbb{E}$ is over the training data $(X_1, Y_1), \ldots, (X_n, Y_n)$ as well as the test data (X_{new}, Y_{new}). This generalization error avoids the fact that it would get smaller the more complex the model for $\hat{m}(\cdot)$. In particular, it will increase as $\hat{m}$ is overfitting. Hence the generalization error in (4.1) is a very useful quantity to regularize and tune an estimator or algorithm for $\hat{m}$.

4.3 Constructing training-, test-data and cross-validation

Unfortunately, test data is not available at the time we want to run and tune our estimator or algorithm. But we can always "artificially" construct (smaller) training- and test-data.

4.3.1 Leave-one-out cross-validation

For leave-one-out cross-validation (CV), we use the ith sample point as test data (test sample size = 1) and the remaining $n-1$ sample points as training data.

Denote in general the estimator or algorithm by $\hat{\theta}_n$ which is based on the n sample points. In CV, when deleting the ith sample, the estimator is based on the sample without the ith observation, and we denote this by

$$\hat{\theta}_{n-1}^{(-i)},\ i = 1, \ldots, n.$$

We can then evaluate this estimate on the ith observation (the test sample), for every $i = 1, \ldots, n$. To make this more explicit, we suppose that the estimator $\hat{\theta}$ is a curve estimator $\hat{m}$, and performance is evaluated in terms of a loss function ρ, e.g. $\rho(u) = u^2$ as in Chapter 3. The cross-validated performance is then

$$n^{-1} \sum_{i=1}^{n} \rho\left(Y_i, \hat{m}_{n-1}^{(-i)}(X_i)\right) \tag{4.2}$$

which is an estimate of the test set error, or generalization error, in (4.1).

Note that the CV-method is very general: it can be used for any loss function ρ and in many problems which can be different than linear or nonparametric regression.

CV in general requires that the estimator $\hat{\theta}$ is fitted n-times, namely for all training sets where the ith observation has been deleted ($i = 1, \ldots, n$).

4.3.2 K-fold Cross-Validation

A computationally cheaper version of leave-one-out CV is the so-called K-fold CV. The data set is randomly partitioned into K equally sized (as equal as possible) subsets $\mathcal{B}_k$ of $\{1, \ldots, n\}$ such that $\cup_{k=1}^{K} \mathcal{B}_k = \{1, \ldots, n\}$ and $\mathcal{B}_j \cap \mathcal{B}_k = \emptyset$ ($j \neq k$). We can now set aside a kth test data set including all sample points whose indices are elements of $\mathcal{B}_k$.

K-fold cross-validation then uses the sample points with indices not in $\mathcal{B}_k$ as training set to construct an estimator

$$\hat{\theta}_{n-|\mathcal{B}_k|}^{(-\mathcal{B}_k)}.$$

The analogue of the evaluation formula in (4.2) is then, for regression with $\hat{m}(\cdot)$,

$$K^{-1} \sum_{k=1}^{K} |\mathcal{B}_k|^{-1} \sum_{i \in \mathcal{B}_k} \rho\left(Y_i, \hat{m}_{n-|\mathcal{B}_k|}^{-(\mathcal{B}_k)}(X_i)\right).$$

K-fold CV thus only needs to run the estimation algorithm K times. Leave-one-out CV is the same as n-fold CV. In practice, often $K = 5$ or $K = 10$ are used.

4.3.3 Random divisions into test- and training-data

K-fold CV has the disadvantage that it depends on the **one** realized random partition into subsets $\mathcal{B}_1, \ldots, \mathcal{B}_K$. In particular, if the data (pairs) are assumed to be i.i.d., the indexing of the data (pairs) should not have an influence on validating a performance: note that leave-one-out CV is indeed independent of indexing the data.

In principle, we can generalize leave-one-out CV to leave-d-out CV. Leave a set $\mathcal{C}$ comprising d observations out (set them aside as test data) and use the remaining $n - d$ data points for training the statistical model or algorithm:

$$\hat{\theta}_{n-d}^{(-\mathcal{C})}, \text{ for all possible subsets } \mathcal{C}_k,\ k = 1, 2, \ldots, \binom{n}{d}.$$

We can then evaluate this estimate on observations from the test set $\mathcal{C}_i$ (the test sample), for every i. The analogue of (4.2), for regression with $\hat{m}(\cdot)$, is then

$$\binom{n}{d}^{-1} \sum_{k=1}^{\binom{n}{d}} d^{-1} \sum_{i \in \mathcal{C}_k} \rho\left(Y_i, \hat{m}_{n-d}^{(-\mathcal{C}_k)}(X_i)\right) \tag{4.3}$$

which is also an estimate of the test set error, or generalization error in (4.1).

The computational burden becomes immense if $d \geq 3$. A computational short-cut is then given by randomization: instead of considering *all* subsets (test sets), we draw B **random** test subsets

$$\mathcal{C}_1^*, \ldots, \mathcal{C}_B^* \text{ i.i.d.} \sim \text{Uniform}(\{1, \ldots, \binom{n}{d}\}),$$

where the Uniform distribution assigns probability $\binom{n}{d}^{-1}$ to every possible subset of size d. Such a Uniform distribution, or such a random subset $\mathcal{C}^*$, is easily constructed by **sampling without replacement**:

draw d times randomly without replacement from $\{1, \ldots, n\}$, yielding a subset $\mathcal{C}^*$.

The random approximation for (4.3) is then

$$B^{-1} \sum_{k=1}^{B} d^{-1} \sum_{i \in \mathcal{C}_k^*} \rho\left(Y_i, \hat{m}_{n-d}^{(-\mathcal{C}_k^*)}(X_i)\right). \tag{4.4}$$

For $B = \infty$, the two expressions in (4.4) and (4.3) coincide (note that we only would need a finite, maybe huge, amount of computation for evaluation of (4.3)).

In practice, we typically choose $d = [\gamma n]$ with $\gamma \approx 0.1$ (10% test data). For the number of random test and training sets, we choose $B \approx 50 - 500$, depending on the cost to compute $\hat{\theta}_{n-d}^{(-\mathcal{C})}$ for a training set of size $n - d$ (evaluation on the test set $\mathcal{C}$) is usually fast). Thus, in terms of computation, the stochastic version in (4.4) may be even faster than leave-one-out CV in (4.2) if $B < n$, and we can use the stochastic approximation also for leave-one-out CV when sample size n is large.

4.4 Properties of different CV-schemes

4.4.1 Leave-one-out CV

Leave-one-out CV, which is equal to n-fold CV, is approximately unbiased for the true prediction or generalization error: the only drawback in terms of bias is that we use training sample size $n - 1$ instead of the original n, causing for a slight bias. The variance of leave-one-out CV is typically high, because the n training sets are so similar to each other:

$$\text{Var}\left(n^{-1} \sum_{i=-1}^{n} \rho\left(Y_i, \hat{m}_{n-1}^{(-i)}(X_i)\right)\right) = n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \text{Cov}\left(\rho(Y_i, \hat{m}_{n-1}^{(-i)}(X_i)), \rho(Y_j, \hat{m}_{n-1}^{(-j)}(X_j))\right).$$

Although (X_i, Y_i) is typically assumed to be independent from (X_j, Y_j), the covariances are substantial because of strong correlation of $\hat{m}_{n-1}^{(-i)}(\cdot)$ and $\hat{m}_{n-1}^{(-j)}(\cdot)$, and hence the double sum in the formula above can be quite large.

4.4.2 Leave-d-out CV

Heuristically, it is clear that leave-d-out CV, with $d > 1$, has higher bias than leave-one-out CV; because we use training samples of sizes $n - d$ instead of the original n, causing some bias. In terms of variance, we average over more, although highly correlated summands in (4.3), which can be shown to decrease variance in comparison to leave-one-out CV.

4.4.3 Versions with computational shortcuts

K-fold CV has larger bias than leave-one-out CV; because the training sets are of smaller sample size than $n-1$ (in leave-one-out CV), and also smaller than the original sample size n. In terms of variance, it is not clear (sometimes wrongly stated in the literature) whether K-fold CV has smaller variance than leave-one-out CV.

The stochastic approximation is expected to have somewhat higher bias and variance than the computationally infeasible leave-d-out CV: it is difficult to assess how much we lose by using a finite B.

4.5 Computational shortcut for some linear fitting operators

Consider the special case of fitting a cubic smoothing spline or fitting a least squares parametric estimator: in both cases, we have a linear fitting operator $\mathcal{S}$,

$$(\hat{m}(x_1), \ldots, \hat{m}(x_n))^\top = \mathcal{S}\mathbf{Y},\ \mathbf{Y} = (Y_1, \ldots, Y_n)^\top.$$

When focusing on the squared error loss function $\rho(y,x) = |y-x|^2$, there is a surprising result for representing the leave-one-out CV score in (4.2):

$$n^{-1}\sum_{i=1}^{n}\left(Y_i - \hat{m}_{n-1}^{(-i)}(X_i)\right)^2 = n^{-1}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{m}(X_i)}{1-\mathcal{S}_{ii}}\right)^2. \tag{4.5}$$

The interesting property is that we can compute the CV score by fitting the original estimator $\hat{m}(\cdot)$ **once on the full dataset**, without having to do it n times by holding one observation back as a test point. Moreover, by using efficient linear algebra implementations, the elements $\mathcal{S}_{ii}$ can be computed with $O(n)$ operations.

Historically, it was computationally more difficult to obtain all diagonal elements from the hat matrix $\mathcal{S}_{ii}$ $(i = 1, \ldots, n)$; and computing the trace$(\mathcal{S})$, which equals the sum of the eigenvalues of $\mathcal{S}$, has been easier. The generalized cross-validation was then proposed in the late 70's:

$$\mathrm{GCV} = \frac{n^{-1}\sum_{i=1}^{n}(Y_i - \hat{m}(X_i))^2}{(1-n^{-1}\mathrm{trace}(\mathcal{S}))^2}.$$

This again requires to compute $\hat{m}(\cdot)$ only once on the full dataset. Moreover, if all the diagonal elements $\mathcal{S}_{ii}$ would be equal (which is typically not the case), GCV would coincide with the formula in (4.5). As outlined already, GCV has been motivated by computational considerations which are nowadays not very relevant anymore. However, the statistical properties of GCV can also be as good (and sometimes better or worse) than the ones from leave-one-out CV.