Topic 1.1. The overfitting problem

Dr. Osval Antonio Montesinos López
oamontes2@hotmail.com
Universidad de Colima
México

The problem of overfitting and underfitting

- The *overfitting* phenomenon occurs when the statistical learning model learns the training data set so well that it performs poorly on unseen datasets.

- Overfitting happens when a statistical learning model learns the systematic and noise (random fluctuations) parts in the training data to the extent that it negatively impacts the performance of the statistical learning model on new data.

- This means that the statistical learning model adapts very well to the noise as well as to the signal that is present in the training data.

- The problem is that affect the model's ability to be generalized.

- Overfitting is more common when learning a loss function from a complex statistical learning model (with more flexibility).

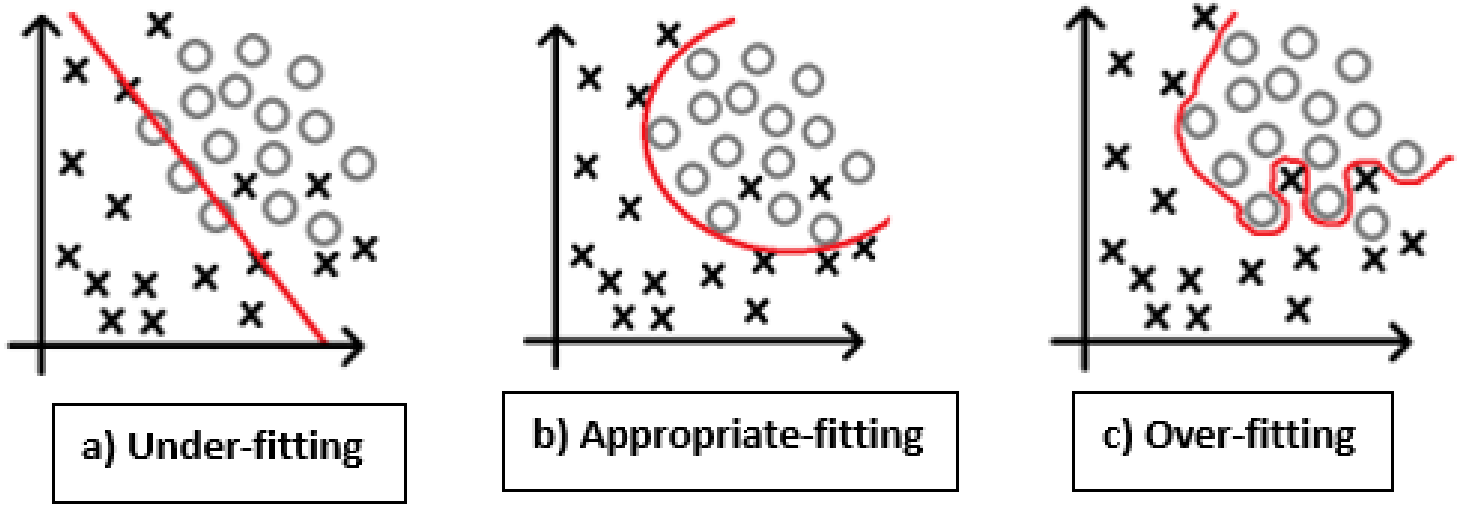
The problem of overfitting and underfitting

- On the other hand, the *underfitted* model occurs when few predictors are included in the statistical learning model (simple model) that poorly represents the complete picture of the predominant data pattern.

- This problem also arises when the training dataset is too small or not representative of the population data.

- An *underfitted* model does not do a good job for fitting the training data and for predicting new data points.

The problem of overfitting and underfitting



We assume that we have three options for the choice of $f(\boldsymbol{x}_i)$: M1, the simple model plotted in Figure panel a; M2, an intermediate model shown in Figure panel b; and M3, a complex model shown in Figure panel c.

The problem of overfitting and underfitting

- In statistical learning the problem of overfitting is not minor and leads to some serious problems in research:

  a) some relationships that seem statistically significant are only noise,

  b) Make really bad prediction on unseen data and,

  c) the model in general is not replicable and predicts badly.

- Finally, according to Shalev-Shwartz and Ben-David (2014), if the learning fails, these are some approaches to follow:

  (1) Increase the sample size of the training set,

  (2) Modify the hypothesis that you have at hand by: (a) enlarging it, (b) reducing it, (c) completely changing it, and (d) changing the parameters you are using;

  (3) Change the feature representation of the data, and

  (4) Change the statistical learning algorithm used

The problem of overfitting and underfitting

Figure 4.10 shows an informal chart that can help us make some decisions while building statistical learning models

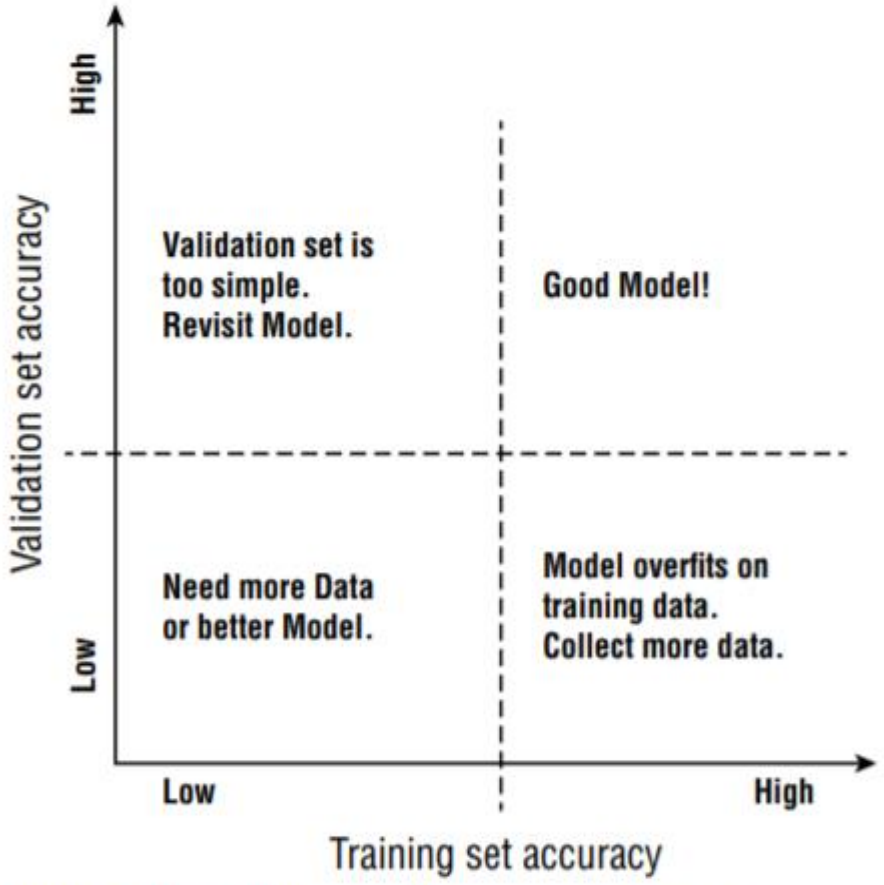


Figure 4.10: Training vs. validation set accuracy

The trade-off between prediction accuracy and model interpretability

- **Accuracy.** It is the ability of a statistical learning model to make correct predictions of unseen data.

- **Flexible models**. These are complex models (SVM, ANN, etc.) which are better in terms of prediction accuracy, but are not interpretable. The bias is reduced, and the variance increases.

- **Inflexible models.** These are simple models (linear regression) are very interpretable but worse in terms of accuracy.

Bias and Variance

- **Bias** is the **difference** between the expected prediction of our statistical learning model and the **true observed values**.

- For example, assume that you poll a specific city where half of the population has high incomes and the other half has low incomes and that you collected a sample of high-income people, you will conclude that the entire city has high income.

- This means that **your conclusion is heavily biased** since you only sampled people with high income.

- On the other hand, **error variance** refers to the amount that **the estimate** of the objective function will **change** using a **different training data set**.

- In other words, the **error variance** accounts for the **deviation of predictions**. Ideally, when a statistical learning model with low error variance predicts a value, the predicted value should remain almost the same even when changing from one training data set to another.

Bias and Variance

- The four combinations of cases resulting from both high and low bias and variance are shown in the Figure 5.2.

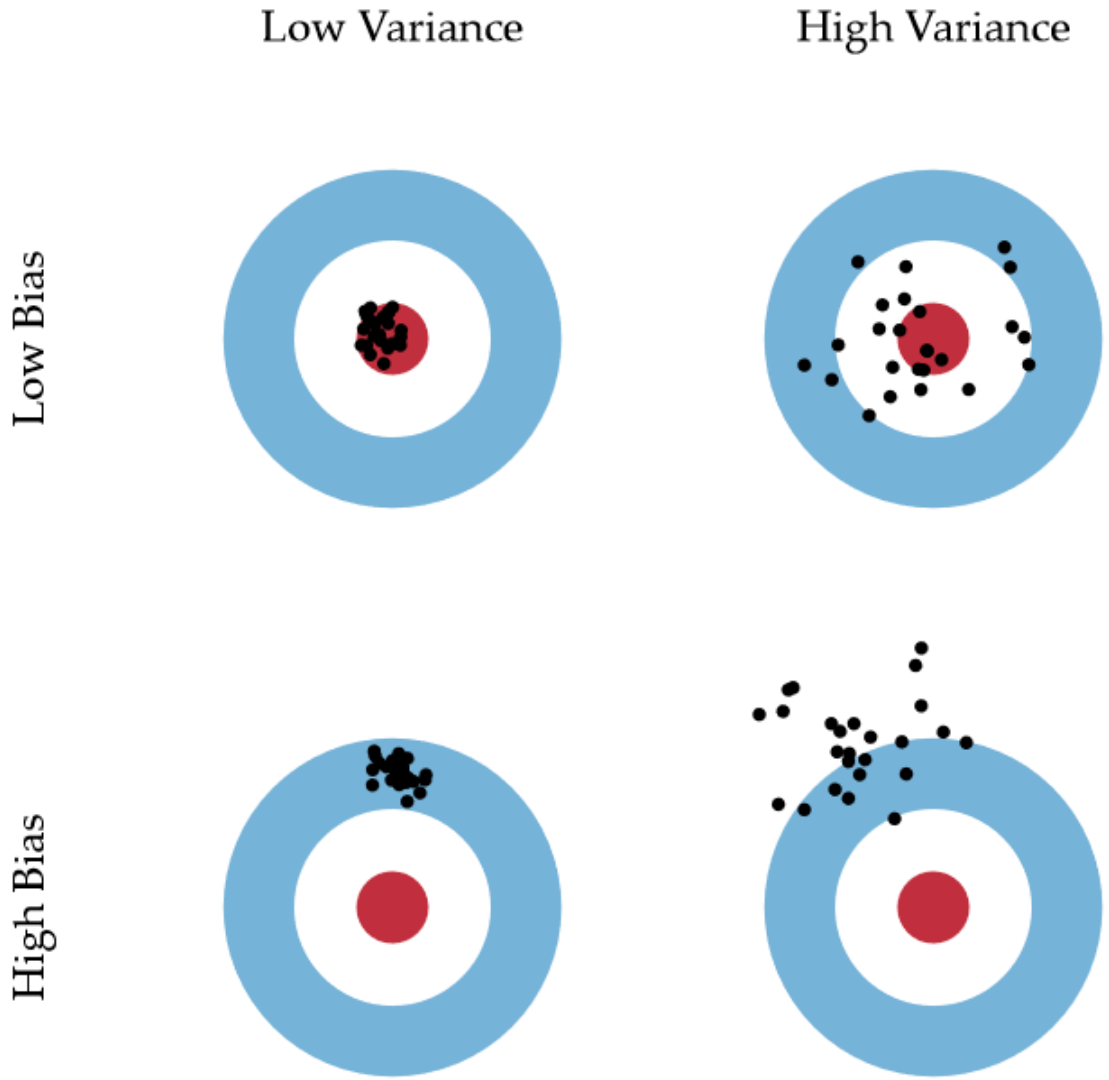


Figure 5.2. Graphical representations of different levels of bias and variance.

Bias and Variance

- However, the election of f^* is desirable to f under the prediction approach. We show this using the statistical model $y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$, which is assumed to be correctly specified with respect to F.
- Using data, we obtain the estimated model $\hat{f}=\hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3$, which has the following properties:
- $Bias = 0$
- $Var(\hat{f}(\boldsymbol{x}_i) = Var\left(\hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3\right) = \sigma^2 \boldsymbol{x^T} (\boldsymbol{X X^T})^{-1} \boldsymbol{x}$
- where $\boldsymbol{x}$ is the vector $\boldsymbol{x}$ $=[x_1, x_1, x_1]^T$ and $\boldsymbol{X}$ is the design matrix based on both predictors. **Combining the squared bias with the variance gives, as expected, the expected** prediction error (EPE).
- $E\left(y - \hat{f}\{\boldsymbol{x}_i\}\right)^2 = \sigma^2 + 0 + \sigma^2 \boldsymbol{x^T} (\boldsymbol{X X^T})^{-1} \boldsymbol{x} = \sigma^2 [1 + \boldsymbol{x^T} (\boldsymbol{X X^T})^{-1} \boldsymbol{x}]$

Bias and Variance

In comparison, consider the estimated underspecified form $\hat{f}^*(\boldsymbol{x}_i) = x_1\hat{\gamma}$. The bias and variance here are:

- $Bias = E(\hat{f}\{\boldsymbol{x}_i\}) - f\{\boldsymbol{x}_i\} = x_1(x_1x_1^T)^{-1}x_1^T(\beta_1x_1 + \beta_2x_2 + \beta_3x_3)$-$(\beta_1x_1 + \beta_2x_2 + \beta_3x_3)$
- $Var(\hat{f}(\boldsymbol{x}_i) = \sigma^2x_1(x_1x_1^T)^{-1}x_1^T$

Combining the squared bias with the variance the EPE is equal to

$$E\left(y - \hat{f}\{\boldsymbol{x}_i\}\right)^2 =$$
$$[x_1(x_1x_1^T)^{-1}x_1^T(x_2\beta_2 + x_3\beta_3) - (x_2\beta_2 + x_3\beta_3)]^2 + \sigma^2[1 + x_1(x_1x_1^T)^{-1}x_1^T]$$

Although the bias of the underspecified model $f^*(\boldsymbol{x}_i)$ is larger than that of $f\{\boldsymbol{x}_i\}$, its variance can be smaller, and in some cases so small that the overall EPE will be lower for the underspecified model.

Bias and Variance

- This means that the underspecified model produces more accurate predictions, in terms of lower EPE, in the following situations:
- (a) when the data are very noisy (large σ^2);
- (b) when the true absolute values of the left-out parameters (in our example β_2 and β_3) are small;
- (c) when the predictors are highly correlated; and
- (d) when the sample size is small or the range of left-out variables is small (Shmueli, 2010).
- Therefore "We note that the practice in applied research of concluding that a model with a higher predictive validity is "truer," is not a valid inference (Hagerty and Srinivasan (1991). Since a parsimonious but less true model can have a higher predictive validity than a truer but less parsimonious model."

Cross-validation

- Cross-validation (CV) is a strategy for model selection or algorithm selection.
- CV consists of splitting the data (at least once) for estimating the error of each algorithm.
- Part of the data (the training set) is used for training each algorithm, and the remaining part (the testing set) is used for estimating the error of the algorithm.
- Then, CV selects the algorithm with the smallest estimated error in new data.
- For this reason, CV is used to evaluate the prediction performance of a statistical learning model in out-of-sample data, that is, that this technique ensures that the data used for training the statistical learning model are independent from the testing data set.
- It consists of repeating and recording the arithmetic average obtained from the evaluation measures on different partitions.

Cross-validation *The single hold-out set approach*

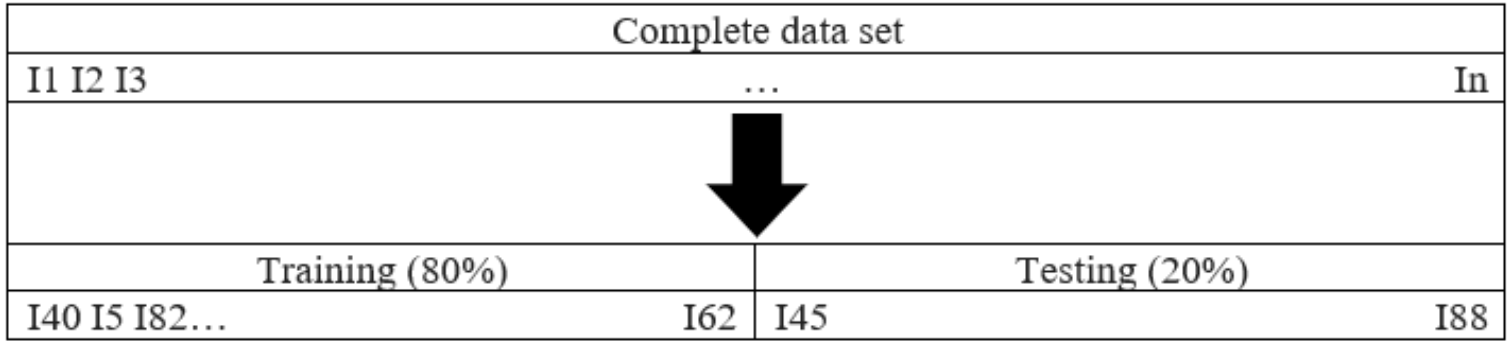


Figure 5.3. Single Hold-out set.

- The single hold-out set or validation set approach consists of randomly dividing the available data set into a training set and a validation set or hold-out set (Figure 5.3).

- One problem of the hold-out (validation) set approach is that it depends on just one training-testing split and its performance depends on how the data are split into the training and testing sets.

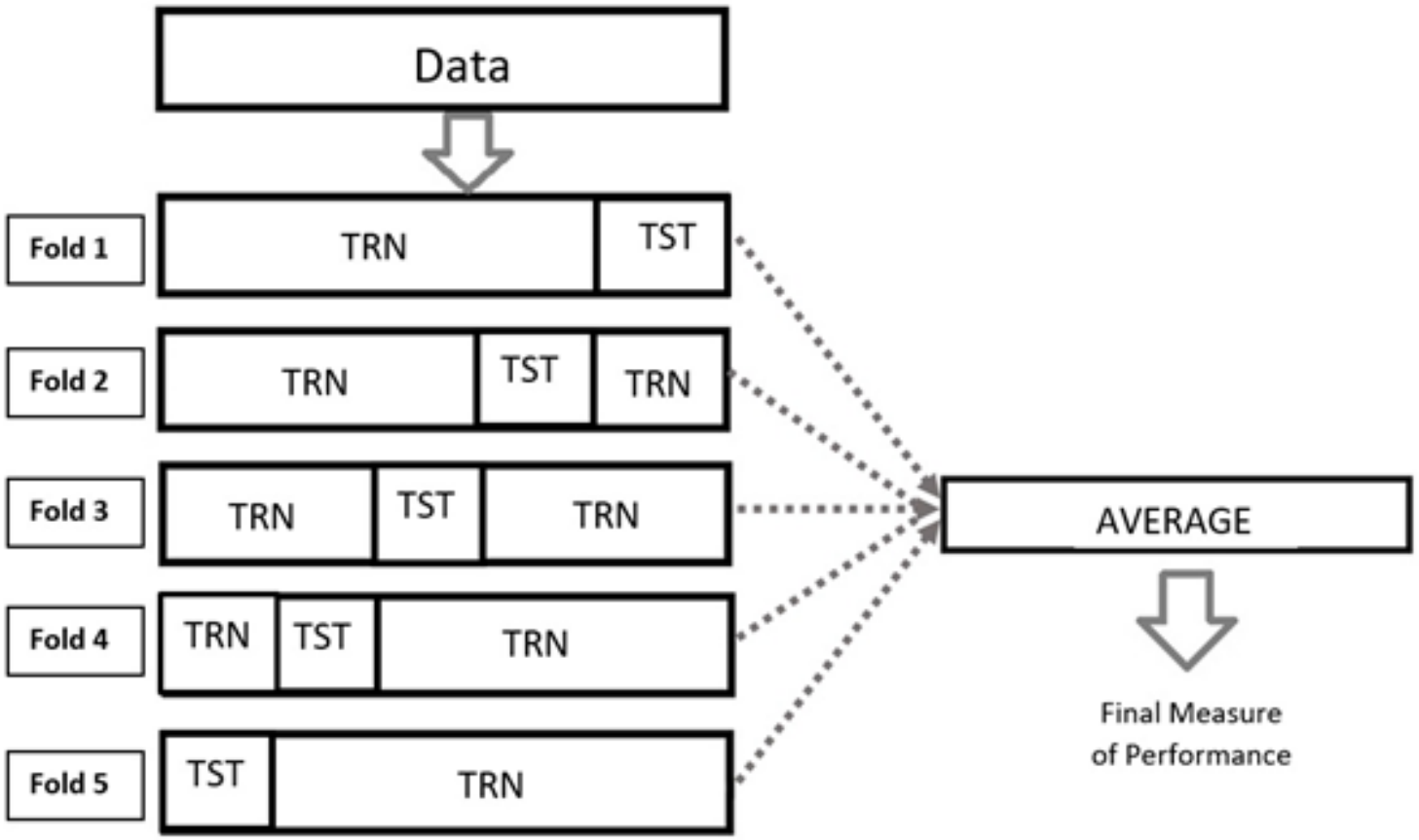
- A set of observations are randomly split into a training set with individuals I40, I5, I82, among others, and into a testing set with observations I45, I88, among others. The statistical learning model is fitted on the training set and its performance is evaluated on the validation set (James et al., 2013).

Cross-validation
The k-fold cross-validation

- In k-fold CV, the data set is randomly divided into k complementary folds (groups) of approximately equal size. One of the subsets is used as testing data and the rest (k-1) as training data.
- Then k-1 folds are used for training the statistical learning model and the remaining fold for evaluating the out of sample prediction performance.
- This method is very accurate because it combines k measures of fitness resulting from the k training and testing data sets into which the original data set was divided, but at the cost of more computational resources.
- In practice, the choice of the number of folds depends on the measurement of the data set, although 5 or 10 folds are the most common choices.

Cross-validation

The k-fold cross-validation



It is important to point out that to reduce variability, we recommend implementing the k-fold CV s times, but each time using different complementary subsets to form the folds.

Cross-validation
The Leave-One-Out cross-validation

- The Leave-One-Out (or LOO) CV is very simple since each training data set is created by including all the individuals except one, while the testing set only includes the individual left out. Thus, for n individuals in the full data set, we have n different training and testing sets.
- This CV scheme does not waste much data as only one individual is removed from the training set.
- Regarding k-fold cross-validation that was just explained, one builds n models from n individuals (samples) instead of k models, where n>k.
- Moreover, each model is trained on n−1 samples rather than (k−1)n/k.

Cross-validation
The Leave-One-Out cross-validation

- In both cases, since k is normally not too large and k<n, LOO is more computationally expensive than k-fold cross-validation.
- In terms of prediction performance, LOO normally produces high variance for the estimation of the test error. Because n−1 of the n samples are used to build each statistical learning model, models constructed from folds are virtually identical to each other and to the model built from the entire training set.
- In general, there is considerable empirical evidence suggesting that 5- or 10-fold cross-validation should be preferred to LOO.

Cross-validation
Random cross-validation

- In this type of CV, the number of partitions (independent training-testing dataset splits) is defined by the user, and the larger, the better.

- Each partition is generated by randomly dividing the whole dataset into two subsets: the training (TRN) dataset and the testing (TST) dataset.

- The percentage of the whole dataset assigned to the TRN and TST datasets is also fixed by the user.
- For example, for each random partition, the user can decide that 80% of the whole dataset can be assigned to the TRN dataset and the remaining 20% to the TST dataset.

Cross-validation
Random cross-validation

- Random cross-validation is different from *k*-fold cross-validation because the partitions are not mutually exclusive; this means that in the random cross-validation approach, one observation can appear in more than one partition.

- Consequently, some samples cannot be evaluated, whereas others can be evaluated more than once, meaning that the testing and training subsets can be superimposed (Montesinos-López et al., 2018a,b).

- To control the randomness for reproducibility, using a specific seed in the random number generator is recommended.

- Using at least 10 random partitions is recommended to get enough accuracy in the estimate of prediction performance.

Cross-validation
The Leave-One-Group-Out cross-validation

- The Leave-One-Group-Out (LOGO) CV is useful when individuals are grouped (in environments or years, or another criterion) and the number of groups (g) is at least two and consist in using as training set the information of g -1 groups and all individuals of the remaining group are used as testing set.

- For example, in the context of genomic selection, many times the same (or different) lines were evaluated in g environments or years, that represent the groups, and the plant breeder is interested in predicting these lines in another environment.

- This type of CV strategy is called CV1 in the context of plant breeding by Jarquin et al. (2017).

Cross-validation
The Leave-One-Group-Out cross-validation

Fold 1	TRN	TST			
Fold 2	TRN		TST		
Fold 3	TRN			TST	
Fold 4	TRN				TST
	Year 1	Year 2	Year 3	Year 4	Year 5

- Under this approach, the predictions are reported for each of the g groups because the scientist is interested in the prediction performance of each environment.
- Many times, the groups are the years under study and the aim is to predict the information of a complete year. However, when the groups are years and if we suspect that there is a considerable correlation between observations that are near in time.
- Therefore, it is very important to evaluate our statistical learning model for time series data on "future" observations. In this case, the training sets are composed of the previous years to predict the following year. See Figure 5.5

Cross-validation
Bootstrap cross-validation

- Bootstrapping is a type of resampling method where, for example, B=10 samples of the same size are repeatedly drawn, with replacement, from a single original sample.
- Then each of these B samples are used to estimate statistics (for example, the mean, variance, median, minimum, etc.) of a population and the average of all the B sample estimates of the target statistic is reported as the final estimate.
- In the context of statistical learning and machine learning, these samples are used to evaluate the prediction performance of the algorithm under study for unseen data.
- One important difference between this CV approach and all the procedures explained above is that now the training set has the same size (number of observations) as the original sample
- As a result, some observations will be represented multiple times in the bootstrap sample, while others will not be selected at all; those observations not selected are referred to as the testing set.

Cross-validation
Bootstrap cross-validation

- Efrom (1983) pointed out that the prediction performance of the bootstrap samples tends to have less uncertainty that the k-fold cross-validation since on average 63.2% of the data points are represented (for training) at least once.

- For this reason, this CV approach has a bias similar to implementing a k=2 fold cross-validation, and the lower the training set, the more problematic the bias.

- To understand this CV method, we provide a simple example of how the training and testing samples are constructed. Assume that we have a sample with 12 individuals denoted as I1, I2, …, I12 and that we will select B=5 bootstrap samples.

Cross-validation

Bootstrap cross-validation

Fold	Training	Testing
1	I7, I4, I6, I9, I2, I3, I4, I4, I8, I6, I8, I7	I1, I5, I10, I11, I12
2	I2, I8, I5, I6, I1, I4, I5, I11, I1 1, I8, I10, I5	I3, I7, I9, I12
3	I5, I9, I11, I3, I10, I5, I7, I2, I3, I11, I6, I9	I1, I4, I8, I12
4	I10, I12, I9, I7, I4, I3, I1, I9, I3, I2, I1, I6	I5, I8, I11
5	I2, I10, I5, I12, I3, I6, I3, I7, I6, I6, I5, I7	I1, I4, I8, I9

Figure 5.6. Schematic representation of bootstrap cross-validation.

Cross-validation
Incomplete block cross-validation

- Incomplete block (IB) CV should be preferred when there are J treatments that were evaluated in I blocks and the same treatments were evaluated in all the blocks.
- The idea behind this CV method is that some treatments should be present in some blocks but absent in others, but the same treatment at least should be present in one environment.
- The theory of incomplete block designs developed in the experimental designs statistical area can be used to construct the training set.
- For example, under a balanced incomplete block (BIB) design, the term incomplete means that all treatments in each block cannot be evaluated, and balanced means that each pair of treatments occur together λ times.

Cross-validation
Incomplete block cross-validation

- The training set is constructed by first defining the % of individuals in the TRN set using the equation sI = Jr =N_{TRN}, where J represent the number of treatments under study, I represents the number of blocks under study, r denotes the number of repetitions of each treatment and s the treatments per block.
- For example, suppose that we had J=10 treatments and I=3 blocks (that is, 30 individuals), and that we decided to use N_{TRN} =21 (70%) of the total individuals in the TRN set.
- Therefore the number of treatments by block can be obtained by solving (sI=N_{TRN}) for s, which results in s=N_{TRN}/I. This means that s=21/3=7 treatments per block.
- Then the corresponding elements for the training set can be obtained with the function find.BIB(10, 3, 7) of the package crossdes of the R statistical software. The numbers used in the function find.BIB() denotes the treatments, the blocks and the treatments per block, respectively. Finally, the treatments that make up the TRN set are shown in Table 5.1.

Cross-validation

Incomplete block cross-validation

Table 5.1. TRN data set for I=3 blocks and J=10 treatments with 70% of data for training.

Block	Treatments per block						
1	1	3	5	7	8	9	10
2	2	3	4	5	6	7	9
3	1	2	4	6	7	8	10

According to Table 5.1 it is clear that each treatment is present in two blocks and missing in one block.

For example, in block 1 the testing set includes treatments 2, 4 and 6,

In block 2 the testing set is composed of treatments 1, 8 and 10,

While for block 3 the testing set is composed of treatments 3, 5 and 9.

Cross-validation

Random cross-validation with blocks

- Random cross-validation with blocks was proposed by López-Cruz et al. (2015) and belongs to the so-called replicated TRN-TST cross-validation that appears in the publication of Daetwyler et al. (2012) since some individuals can never be part of the training set.
- This algorithm, like the incomplete block cross-validation, is appropriate when we are interested in evaluating J lines in I blocks or environments and tries to mimic a prediction problem faced by breeders in incomplete field trials where lines are evaluated in some, but not all, target environments.
- The algorithm for constructing the TRN-TST sets is described in the following steps:
- Step 1. Calculate the total number of observations under study as $N = J \times I$.

Cross-validation

Random cross-validation with blocks

- Step 2. Define the proportion of observations used for training and testing, that is, P_{TRN} and P_{TST}.
- Step 3. Calculate the size of the testing set $N_{TST} = N \times P_{TST}$.
- Step 4. Choose N_{TST} lines at random without replacement if $J \geq N_{TST}$, and with replacement otherwise.
- Step 5. Each chosen line will then be assigned to one of the I environments chosen at random without replacement. All the selected lines and environments will form the training set, while the lines and environments that were not chosen will form the corresponding testing set.
- Step 6. Then steps 1-5 are repeated depending on the number of TRN-TST partitions required, that is, specified a priori (López-Cruz et al., 2015).

Cross-validation
Random cross-validation with blocks

- Next we assume that we have 10 lines or treatments and three environments or blocks and we will form the corresponding training testing sets for only one partition.
- Step 1. The total number of observations under study is $N = 10 \times 3 = 30$.
- Step 2. We define $P_{TRN} = 0.7$ and $P_{TST} = 0.3$.
- Step 3. The size of the testing set is $N_{TST} = 30 \times 0.3 = 9$.
- Step 4. Since $J = 10 \geq N_{TST} = 9$, we selected at random without replacement the following lines: L1, L2, L3, L4, L5, L7, L8, L9 and L10.
- Step 5. Then each chosen line was assigned to one of the $I = 3$ environments randomly chosen without replacement, as shown in Table 5.2.

Cross-validation

Random cross-validation with blocks

Table 5.2. TRN-TST data sets for $J = 10$ lines and $I = 3$ environements.

Environment	Lines									
	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
1	TRN	TST	TRN	TRN	TST	TRN	TRN	TST	TRN	TRN
2	TST	TRN	TRN	TRN	TRN	TRN	TST	TRN	TST	TRN
3	TRN	TRN	TST	TST	TRN	TRN	TRN	TRN	TRN	TST

Cross-validation
Other options and general comments on cross-validation

- It is important to point out that when the data set is considerably large, it is better to randomly split the data set into three parts: a training set, a validation set, and a testing set.
- The training set and testing set are used as explained before, while the validation set is used to estimate the prediction error for model selection,
- Understanding by model selection the process of estimating the performance of different models in order to choose the best one.

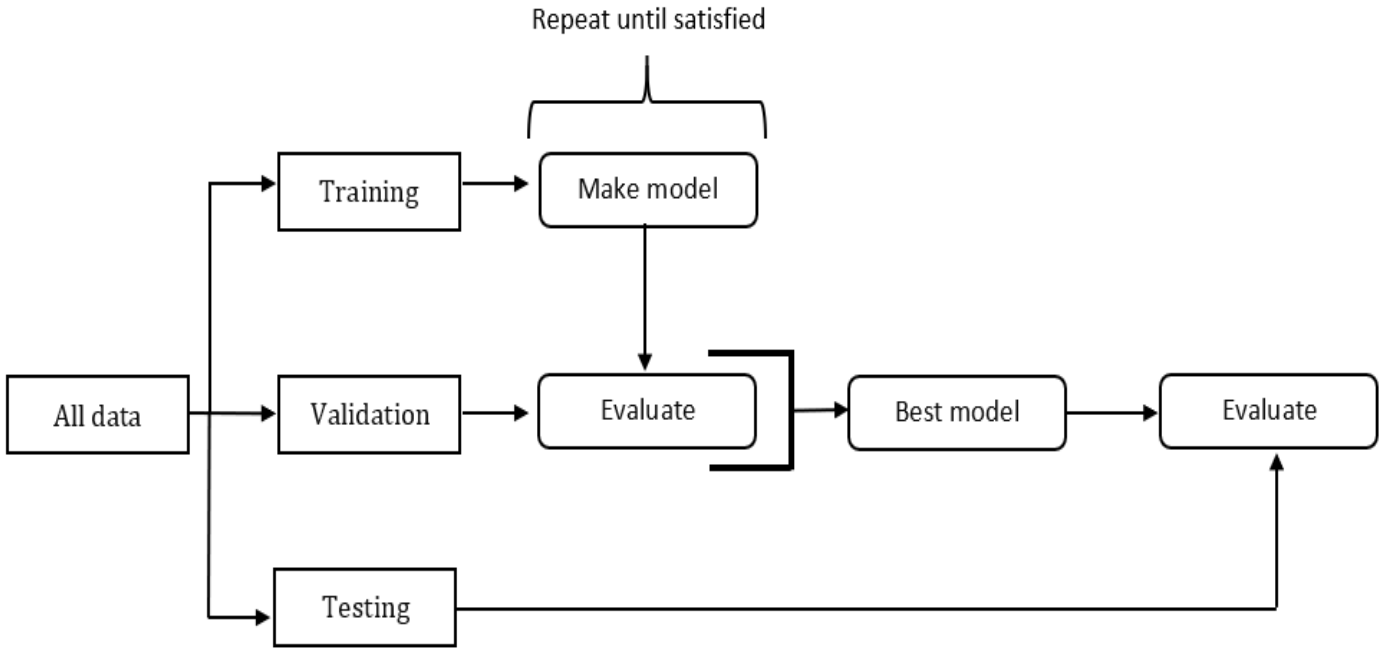


Figure 5.7. Schematic representation of the training, validation and testing sets proposed by Cook (2017).

Cross-validation

Other options and general comments on cross-validation

- For example, assume that our data set has 50,000 rows (observations) and that we decided to use 5000 rows for the testing set, and 5000 rows for the validation set. This means that 40,000 rows are left for the training set.
- Then we train our statistical learning model with each component of the grid of hyperparameters of the training set, and we evaluated the prediction performance on the validation set, as shown in the middle of Figure 5.7.
- Finally, we picked the best model (best subset of hyperparameters) in terms of prediction performance in the validation set and we are ready to evaluate the prediction performance in the testing set.
- Finally, we report the testing error on the testing set, as can be observed at the bottom of Figure 5.7 (Cook, 2017).

Cross-validation

Other options and general comments on cross-validation

- Under this approach, the testing and validation sets have approximately the same size to guarantee a similar out-of-sample prediction performance. Since it is difficult to give general rules on how to choose the number of observations in each of the three parts, a typical number might be 50% for training, and 25% each for validation and testing (Hastie et al., 2008) or 70%, 15% and 15% for training, validation and testing, respectively.

- Another example for finding the optimal setting of hyperparameters using the grid search that is very common in deep learning consists of picking a value for each parameter from a finite set of options [e.g., number of epochs (100, 150, 200, 250, 300), batch sizes (25, 50, 75, 100, 125), number of layers (1,2,3,4,5), and types of activation functions (RELU, Sigmoid), …] and training the statistical learning model with every permutation of hyperparameter choices using the training set.

Cross-validation
Other options and general comments on cross-validation

- It is important to point out that when you have more than one random partition (training-testing-validation), as shown in Figure 5.8, the same process provided in Figure 5.7 is followed but the average of all the partitions is reported as a measure of prediction performance. Also, if you want more precision in the estimated prediction performance, you can repeat the process given in Figure 5.8 many times and report the average of all repetitions as a measure of prediction performance.

Then we chose the combination of hyperparameters with the best prediction performance on the validation set, and we report the prediction performance of the best selected model (set of hyperparameters) in the testing set (Buduma, 2017).

Figure 5.8. Five partitions of training-validation-testing.

Partition 1	TRN			TST	VAL
Partition 2	TRN		TST	VAL	TRN
Partition 3	TRN	TST	VAL	TRN	
Partition 4	TST	VAL	TRN		
Partition 5	VAL	TRN			TST

Cross-validation

Other options and general comments on cross-validation

- As mentioned above, this approach (Figures 5.7 and 5.8) is used when the amount data is large; however, when the amount data is not really large, modifying this approach is suggested to avoid wasting too much training data in validation sets.
- Outer-CV and Inner-CV. This modification consists of performing an inner cross-validation approach since the training set is split into complementary subsets, and each model is trained against a different combination of these subsets and validated against the remaining parts.
- Once the model hyperparameters have been selected, a final model is trained with the whole training set (refitted) and the generalized prediction performance is measured on the testing set.

Cross-validation

Other options and general comments on cross-validation

- This approach was applied by Montesinos-López et al. (2018a, b) but they split the original data into training and testing sets. Then each training set was split again and 80% of the data was used for training a grid of hyperparameters and the remaining 20% was used for testing the prediction performance and selecting the best combination of hyperparameters with the best prediction performance. Then they refitted the deep learning algorithm with the whole training set, and with this they evaluated the out-of-sample prediction.

- They called the conventional training-testing partition outer cross-validation, while the split performed in each training set used for hyperparameter tuning they called inner-cross validation. Finally, it is important to point out that any type of the cross-validation strategies mentioned in this section (random CV, k-fold CV, Bootstrap CV, etc., IB CV) can be used in both outer and inner CV. For example, you can use a 5-fold CV in both outer and inner CV.

Model tuning

- A hyperparameter is a parameter whose value is set before the learning process begins.
- Hyperparameters govern many aspects of the behavior of statistical learning models, since different hyperparameters often result in models with significantly different performance.
- This means that tuning hyperparameter values is a critical aspect of the statistical model training process and a key element for the quality of the resulting prediction accuracies.
- However, a good choice of hyperparameters is challenging (Montesinos-López et al., 2018a).

Model tuning

- Hyperparameter tuning finds the best version of a statistical learning model by running many training sets on a ranges of values of hyperparameters that you specify.
- Then the hyperparameter values that provide the best performance in out-of-sample prediction is selected.
- There are many ways of searching for the best hyperparameters.
- A schematic representation of the tuning process proposed by Kuhn and Johnson (2013) is given in Figure 5.9.
- It is important to point out that this process should be performed correctly because when the same data are used for training and evaluating the prediction performance, the prediction performance obtained is extremely optimistic.

Model tuning

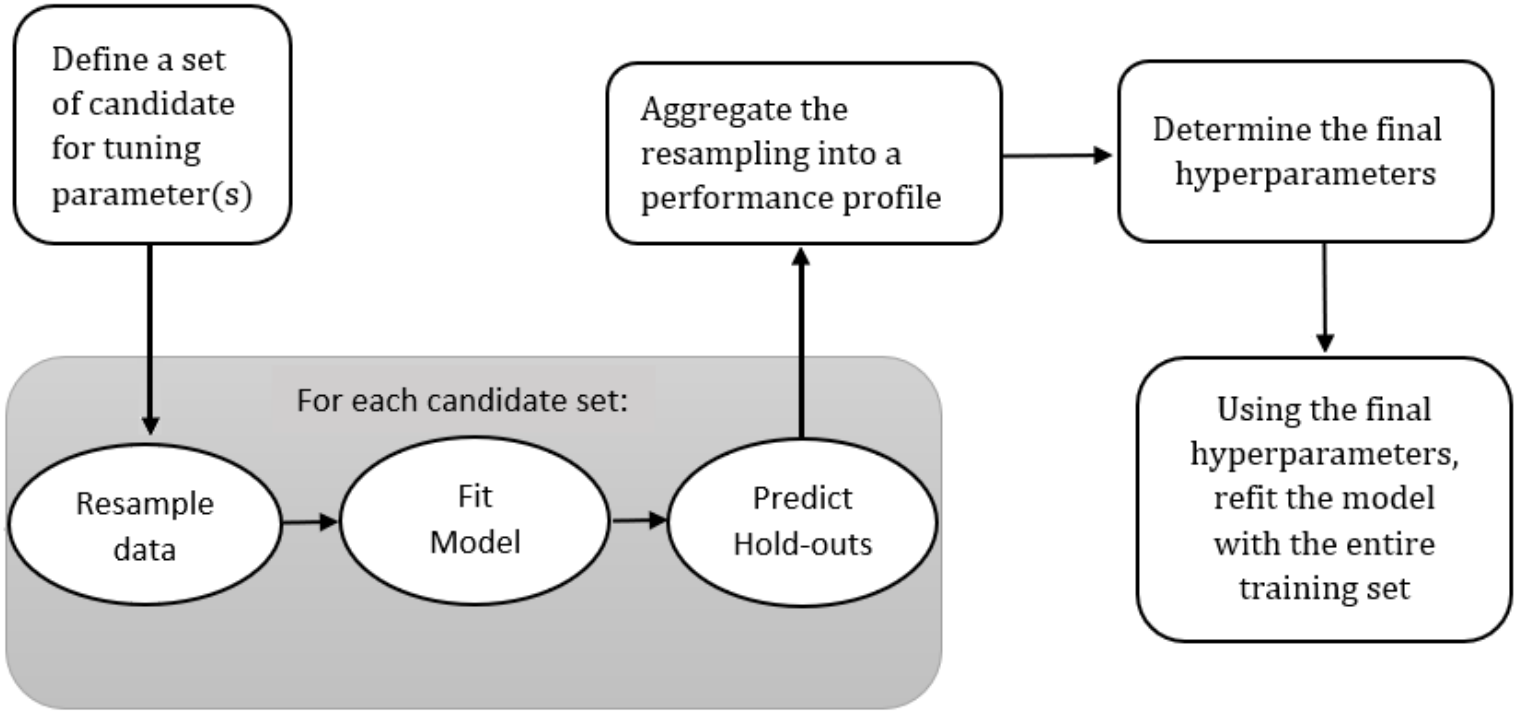


Figure 5.9. Schematic representation of the tuning process proposed by Kuhn and Johnson, (2013).

Model tuning
Why is model tuning important?

- Tuning the hyperparameters of the models is a key element to optimize your statistical learning model to perform well in out-of-sample predictions.
- The tuning process is more an art than a science because there is no unique formal scientific procedure available in the literature.
- Nowadays the tuning process is like a trial-and-error process that consists of implementing the statistical learning model many times with different values of the hyperparameters and then comparing its performance on your testing set in order to determine which set of hyperparameters results in the most accurate model, and for the final implementation the set of hyper-parameters of the best model is used.

Model tuning

Methods for hyperparameter tuning (grid search, random search, etc.)

- Manual tuning of statistical learning models is of course possible but relies heavily on the user's expertise and understanding of the underlying problem.
- Some approaches for hyperparameter tuning reported in the literature are:

  (a) grid search,

  (b) random search,

  (c) Latin hypercube sampling, and

  (d) optimization (Koch et al., 2017).

Model tuning

Methods for hyperparameter tuning (grid search, random search, etc.)

- For example, in Ridge regression, this approach is implemented as follows: since λ is the hyperparameter to be tuned, first we propose, for example, a grid of 100 values for this hyperparameter from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$.
- Then we divide the training set into 5 inner training sets and 5 inner testing sets and each of the 100 values of the grid is fitted using the inner training sets and the testing error is evaluated with the inner testing sets.
- Then we get the average predicted test error and we pick one value out of the 100 values of the grid that produces the best prediction performance.

Model tuning

Methods for hyperparameter tuning (grid search, random search, etc.)

- Next, we refit the statistical learning method to the whole training set using the picked value of λ, and finally we perform the predictions for the testing set using the learned parameters of the training set with the best picked value of λ.

- In all those models with one hyperparameter, it is practical to implement the *grid search* method, but for example, in deep learning models that many times require 6 hyperparameters to be tuned, if only three values are used for each hyperparameter, there are $3^6 = 729$ combinations that need to be evaluated, which becomes computationally impracticable.

Model tuning

Methods for hyperparameter tuning (grid search, random search, etc.)

- A *random search* differs from a *grid search* in that we no longer provide a discrete set of values to explore for each hyperparameter; rather, we provide a statistical distribution for each hyperparameter from which values may be randomly sampled. This allows a much greater chance of finding effective values for each hyperparameter.

- Optimization methods, on the other hand, consist of sequential model-based optimization that allows using the results of previous experiments to improve the sampling method of the next experiment. However, the implementation of these optimization methods is not straightforward because it requires expensive computation;

- Finally, although it is challenging, the tuning process often leads to hyperparameter settings that are better than the default values.