WDS'10 Proceedings of Contributed Papers, Part I, 31–36, 2010. ISBN 978-80-7378-139-2 © MATFYZPRESS

Data Splitting

Z. Reitermanová

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.

Abstract. In machine learning, one of the main requirements is to build computational models with a high ability to generalize well the extracted knowledge. When training e.g. artificial neural networks, poor generalization is often characterized by over-training. A common method to avoid over-training is the hold-out cross-validation. The basic problem of this method represents, however, appropriate data splitting. In most of the applications, simple random sampling is used. Nevertheless, there are several sophisticated statistical sampling methods suitable for various types of datasets.

This paper provides a survey of existing sampling methods applicable to the data splitting problem. Supporting experiments evaluating the benefits of the selected data splitting techniques involve artificial neural networks of the back-propagation type.

Introduction

In machine learning, one of the main requirements is to build computational models with high prediction and generalization capabilities [Mitchell, 1997]. In the case of supervised learning, a computational model is trained to predict outputs of an unknown target function. The target function is represented by a finite training dataset T of examples of inputs and the corresponding desired outputs: $T = \{ [\vec{x}_1 , \vec{d}_1], \ldots , [\vec{x}_n , \vec{d}_n] \}$, where $n > 0$ is the number of ordered pairs of input/output samples (patterns).

At the end of the training process, the final model should predict correct outputs for the input samples from T, but it should also be able to generalize well to previously unseen data. Poor generalization can be characterized by over-training. If the model over-trains, it just memorizes the training examples and it is not able to give correct outputs also for patterns that were not in the training dataset. These two crucial demands (good prediction on T and good generalization) are conflicting and are also known as the *Bias and Variance dilemma* [Kononenko and Kukar, 2007].

A common technique to balance between minimal *Bias* and minimal *Variance* of the model is the cross-validation. The basic problem of this technique represents, however, appropriate data splitting. Improper split of the dataset can lead especially to an excessively high *Variance* of the model performance. However, various sophisticated sampling methods can be used to deal with this problem.

In the first section of this paper, the cross-validation techniques and the problem of data splitting are described. The second section reviews the sampling methods, which can be used to solve this problem. The discussed methods are ordered by their algorithmic complexity. The third section contains analysis of the discussed methods and results of the experimental comparison. Supporting experiments evaluating the benefits of the selected data splitting techniques involve artificial neural networks of the back-propagation type.

Cross-validation techniques

Cross-validation techniques [Refaeilzadeh et al., 2009; Picard and Cook, 1984] belong to conventional approaches used to ensure good generalization and to avoid over-training. The basic idea is to divide the dataset T into two subsets – one subset is used for training while the other subset is left out and the performance of the final model is evaluated on it. The main purpose of cross-validation is to achieve a stable and confident estimate of the model performance. Cross-validation techniques can also be used when evaluating and mutually comparing more models, various training algorithms, or when seeking for optimal model parameters [Reed and Marks, 1998].

This article is focused on the two most commonly used types of cross-validation – hold-out cross-validation (early stopping) and k-fold cross-validation.

Hold-out cross-validation (early stopping)

Hold-out cross-validation is a widely-used cross-validation technique popular for its efficiency and easiness. It separates the dataset T (of size n) into three mutually disjoint subsets – training T_{tr}, validation T_v, and testing T_t of sizes n_{tr}, n_v and n_t successively. An advantage of this method is, that the proportion of these three data subsets is not strictly restricted. The model is trained on the training subset T_{tr}, while the validation subset T_v is periodically used to evaluate the model performance during the training to avoid over-training. The training is stopped, when the performance on T_v is good enough or when it stops improving. When mutually comparing $m > 1$ computational models $L_1, \cdots, L_m$ against each other, the testing subset T_t is used to gain a confident estimate of the models' performance. Algorithm 1 describes this process in more detail.

Algorithm 1 Hold-out cross-validation

```
1. Input: dataset $T$, performance function $error$, computational models $L_1, \cdots, L_m, m \geq 1$
2. Divide $T$ into three disjoint subsets $T_{tr}$ (training), $T_v$ (validation), and $T_t$ (testing).
3. For $j = 1, \cdots, m$:
   3.1. Train model $L_j$ on $T_{tr}$ and periodically use $T_v$ to asses the model performance:
        $E_v^j = error(L_j(T_v))$.
   3.2. Stop training, when a stop-criterion based on $E_v^j$ is satisfied.
4. For $j = 1, \cdots, m$, evaluate the performance of the final models on $T_t$: $E_t^j = error(L_j(T_t))$.
```

An important question related to the hold-out cross-validation is how to split T into the three subsets. This problem is often denoted as *data splitting* [May et al., 2010]. The chosen split heavily affects the quality of the final model. The estimate of the model performance (evaluated on the testing set) should be stable – it should have a low bias and variance. If the dataset is split poorly, the data subsets will not sufficiently cover the data and especially the variance will increase.

K-fold cross-validation

The k-fold cross-validation [Mitchell, 1997] uses a combination of more tests to gain a stable estimate of the model error. It is useful if not enough data for the hold-out cross-validation is available. The dataset T is divided into k parts of the same size. One part forms the validation (testing) set T_v, the other parts form the training set T_{tr}. This process is repeated for each part of the data. Algorithm 2 describes this method in more detail.

Algorithm 2 K-fold cross-validation

```
1. Input: dataset $T$, number of folds $k$, performance function $error$, computational models
          $L_1, \cdots, L_m, m \geq 1$
2. Divide $T$ into $k$ disjoint subsets $T_1, \cdots, T_k$ of the same size.
3. For $i = 1, \cdots, k$:
   $T_v \leftarrow T_i, T_{tr} \leftarrow \{T \setminus T_i\}$.
   3.1. For $j = 1, \cdots, m$:
        Train model $L_j$ on $T_{tr}$ and periodically use $T_v$ to asses the model performance:
        $E_v^j(i) = error(L_j(T_v))$.
        Stop training, when a stop-criterion based on $E_v^j(i)$ is satisfied.
4. For $j = 1, \cdots, m$, evaluate the performance of the models by: $E_v^j = \frac{1}{k} \cdot \sum_{i=1}^{k} E_v^j(i)$.
```

On the contrary to the previous method, there is not a separate testing set and proportion of the training and validation subsets is strictly restricted by the number of folds k. An important question is how to choose k. In most applications $k = 10$ is chosen. Another important question is how to split the samples into the k subsets – and this is just an extension of the *data splitting* problem described above.

Data splitting

The problem of appropriate data splitting can be handled as a statistical sampling problem. Therefore, various classical statistical sampling techniques can be employed to split the data [May et al., 2010; Lohr, 1999]. These sampling methods can be divided into the following categories based on their principles, goals and algorithmic and computational complexity:

- Simple random sampling (SRS),
- Trial-and-error methods,
- Systematic sampling,
- Convenience sampling,
- CADEX, DUPLEX,
- Stratified sampling.

Some of the methods are simple and widely used, although they suffer from high variance of the model performance (e.g. SRS and trial-and-error methods). Other techniques are deterministic and effective, however restricted to specific types of datasets (e.g. convenience and systematic sampling). The more sophisticated methods (e.g. CADEX, DUPLEX and stratified sampling) exploit the structure of the data to reach confident results at the expense of higher computational costs.

In the following paragraphs, each approach will be described in more detail, while their advantages and drawbacks will be emphasized with respect to the character of the given data. We will start with the most commonly used simple techniques and then proceed to the more sophisticated approaches suitable for complex high-dimensional datasets.

Simple random sampling (SRS)

Simple random sampling is the most common method, it is efficient and easy to implement. Samples are selected randomly with a uniform distribution. E.g. for the training subset T_{tr}: $p(x \in T_{tr}) = \frac{n_{tr}}{n}, n = |T|, n_{tr} = |T_{tr}|$ – each sample has an equal probability of selection. Advantage of this method is, that it leads to low bias of the model performance [Lohr, 1999]. However, for more complex (non-uniformly distributed) datasets or if $n_{tr} << n$, the random selection can easily lead to subsets that don't cover the data properly and therefore the estimate of the model error will have a high variance.

Trial-and-error methods

Trial-and-error (generate and test) methods try to overcome the high variance of the model performance when using SRS by repeating the random sampling several times and then averaging the results. This is time-consuming. More sophisticated techniques try to search the space of potential splits to minimize the statistical difference between T and its subsets. Various criteria can be used such as mean and variance [Bowden et al., 2002]. Distribution of each variable can be considered separately [Bowden et al., 2005]. The main drawback of these methods are the high computational costs and vague theoretical background.

Systematic sampling

Systematic sampling is a deterministic approach designated for naturally ordered datasets, such as time series [Zhang and Berardi, 2001]. At first, a proper ordering of the dataset T has to be found. For the ordered dataset, a random starting sample is chosen and then each k-th sample is taken for $k = \frac{n_{tr}}{n}, n = |T|, n_{tr} = |T_{tr}|$. Systematic sampling is a very efficient method and it is easy to implement. However, for most types of datasets (e.g. multimedia data, gene sequences,..), it is very difficult to find an appropriate ordering. One possibility is to use the distribution of the output variable [Baxter et al., 2000], which may lead to good results if the target function is uniquely invertible. For misordered data, the results of systematic sampling are comparable to the SRS and suffer from the same problems (high variance and also bias of the model performance). Another drawback of systematic sampling is its sensitivity to periodicities in the dataset [Lohr, 1999].

Convenience sampling

Convenience sampling is an efficient deterministic method widely used when dealing with time series [Bowden et al., 2002; Zhang and Berardi, 2001]. The dataset T is split according to discrete blocks, e.g. time intervals. Similarly to the systematic sampling, convenience sampling is advantageous only for datasets of a special type – T must consist of several similarly distributed segments. For datasets that can't be divided into meaningful separate blocks, convenience sampling is at best comparable to SRS. This method is also vulnerable to periodicities, long time trends and changes of the conditions in time [Bowden et al., 2002].

CADEX, DUPLEX

CADEX [Kennard and Stone, 1969] and its extension DUPLEX [Snee, 1977] are methods that select samples based on their mutual Euclidean distance. They start with two most distanced samples from the dataset T and then repetitively select samples with maximal distance to the previously sampled examples. Algorithm 3 describes DUPLEX in more detail. These methods ensure a maximum coverage of T. Unfortunately, enormous computational complexity prevents them to be used for large high-dimensional datasets.

Algorithm 3 DUPLEX

```
1. Input: dataset $T$
2. For each $T_i \in \{T_{tr}, T_v, T_t\}$ :
   2.1. Find $x_j, x_k \in T$ with the maximum mutual distance $\| x_j - x_k \|$.
   2.2. Extract $x_j, x_k$ from $T$ and add them to $T_i$.
3. Repeat:
   For each $T_i \in \{T_{tr}, T_v, T_t\}$ :
   3.1. Find $x \in T$ with the maximum distance $\| x - s \|$ to the nearest previously sampled $s \in T_i$.
   3.2. Extract $x$ from $T$ and add it to $T_i$.
```

Stratified sampling

The basic idea of the stratified sampling is to explore the internal structure and distribution of the dataset T and exploit it to divide T into relatively homogeneous groups of samples (strata, clusters). The samples are then selected separately from each cluster.

Stratified sampling is advantageous for the datasets, which can be divided into separate clusters [May et al., 2010]. In such case, all regions of the input space may be adequately covered by the training subset T_{tr} and the estimate of the model performance measured on T_t will be highly precise. On the other hand, for nearly uniformly distributed datasets, the stratified sampling will be comparable to SRS.

Various clustering algorithms can be used to divide T into clusters including c-means [Fernandes et al., 2008], fuzzy c-means [Kaufman and Rousseeuw, 1990] and self-organizing maps [May et al., 2010]. However, most of the clustering algorithms are sensitive to the choice of initial parameters, such as the desirable number of clusters or the choice of input variables [Oja, 2002]. An inadequate setup of these parameters affects heavily the quality of the sampling.

Another important question is, how to select samples from each cluster. There are two most common principles – a) select one sample from each cluster [Bowden et al., 2002], b) randomly divide each cluster into the subsets [Kingston, 2006; May et al., 2010]. The second approach is often denoted as the stratified random sampling. In the following paragraph, this technique will be described in more detail.

Stratified random sampling

When using the stratified random sampling method, samples from each cluster are selected with a uniform probability. There are more possible ways, how to choose the number of samples to be selected per cluster, the so-called *quota* (n_q). *Quota* can be determined by one of the allocation rules [Cochran, 1977]:

- Equal allocation,
- Proportional allocation,
- Optimal allocation.

Equal allocation rule takes the same number of samples from each cluster – e.g. for the training subset T_{tr}: $n_q = \frac{n_{tr}}{Q}$, where $n_{tr} = |T_{tr}|$ and Q is the number of clusters. This rule is however useless if there are clusters with not enough samples. Proportional allocation rule tries to overcome this problem by setting $n_q = \frac{n_{tr}}{n} \frac{N_q}{\sum_{j=1}^{Q} N_j}$, where $n_{tr} = |T_{tr}|, n = |T|$ and N_j is the size of cluster j. In this case, *quota* depends on the size of each cluster. When applying the Optimal allocation rule, *quota* depends not only on the size N_q of cluster q, but also on the standard deviation σ_q of the samples in cluster q: $n_q = \frac{n_{tr}}{n} \frac{N_q \sigma_q}{\sum_{j=1}^{Q} N_j \sigma_j}$. Therefore, also the width of the clusters is exploited and *quota* is higher for sparse regions of the input space [May et al., 2010].

Analysis of the discussed methods

In the previous section, various sampling methods that can be adopted to solve the data-splitting problem were described. The above-discussed approaches differ in many ways (e.g. algorithmic and computational complexity, necessary requirements on the dataset). Some of the methods are deterministic (e.g. convenience and systematic sampling), other methods are stochastic (e.g. SRS and stratified random sampling).

Each of the described methods has its advantages but also its limitations. The choice of the optimal sampling method for the given problem depends especially on the character of the dataset and on the desired proportion of the subsets. For simple, nearly uniformly distributed datasets, the simple yet efficient methods (e.g. SRS) will be sufficient. For the naturally well-ordered datasets (e.g. time series), even the highly efficient deterministic approaches (e.g. convenience and systematic sampling) achieve good results. However, when dealing with complex high-dimensional datasets, the sophisticated sampling techniques (e.g. CADEX, DUPLEX and stratified sampling) can significantly reduce the bias and variance of the model error. Even though more sophisticated methods reach better results, precision is usually compensated by high computation complexity (e.g. for CADEX and DUPLEX) or by a high sensitivity to the choice of initial parameters (e.g. stratified random sampling). Maybe therefore most of the real-world applications prefer simpler and more efficient methods, such as SRS.

Experimental results

Our experiments involved data splitting methods in the case of hold-out cross-validation. Three of the sampling methods, which represent the most widely applicable approaches, were implemented – the basic method Simple random sampling (SRS) and the two sophisticated methods - DUPLEX and Stratified random sampling with c-means clustering and proportional allocation rule (CRS). As the reference computational model, artificial neural networks of the back-propagation type (ANN) were used. To evaluate the benefits of the selected data splitting techniques on various types of data, three referential datasets were chosen – with ascending size, complexity and number of dimensions:

- $D1$ – computer-generated 7-dimensional dataset of 500 samples (mixture of Gaussians), classification into 2 classes,
- WB – real 30-dimensional dataset of 972 samples (economical indicators), classification into 5 classes,
- PIC – real 113-dimensional dataset of 4000 samples (features extracted from photographs), classification into 3 classes.

For each of the datasets, the three alternative sampling methods were used to split the data into the training (T_{tr}), validation (T_v) and testing (T_t) subsets of the proportion $50\% - 25\% - 25\%$. Then the hold-out cross-validation was used to train the ANN-model and evaluated its performance on the testing subset T_t. Also the computational costs of each of the three sampling techniques were measured. The experiment was repeated 100-times on Core i7 860 with 4GB RAM.

Table 1. The performance of the ANN-model for the alternative sampling techniques. The performance was evaluated on the testing subsets (T_t) using two standard performance measures – the mean-squared error $MSE(T_t)$ and the classification error $CE(T_t)$.

Method	$MSE(T_t)$			$CE(T_t)$		
\ Dataset	D1	WB	PIC	D1	WB	PIC
SRS	0.072 ± 0.020	0.116 ± 0.042	0.479 ± 0.011	0.021 ± 0.008	0.055 ± 0.026	0.225 ± 0.007
DUPLEX	0.073 ± 0.020	0.102 ± 0.031	0.447 ± 0.017	0.022 ± 0.009	0.046 ± 0.021	0.202 ± 0.011
CRS	0.072 ± 0.019	0.100 ± 0.030	0.444 ± 0.018	0.021 ± 0.008	0.045 ± 0.019	0.202 ± 0.009

Table 1 summarizes the results of our experiments. For the simple low-dimensional dataset $D1$, the performance of the ANN-model is similar for all of the three sampling methods. Nevertheless, for the two more complex datasets (WB and PIC), DUPLEX and CRS achieved better results than SRS. The computational costs of the methods are presented in Table 2. For all of the three datasets, SRS is the fastest method. CRS is also relatively fast, while the time complexity of DUPLEX is – especially for the largest dataset PIC – extremely high.

Table 2. The speed of sampling (in seconds) of the alternative sampling techniques averaged over the series of 100 tests.

Method	**Dataset**		
	D1	WB	PIC
SRS	< 0.01	< 0.01	< 0.01
DUPLEX	1.09	1.21	205.5
CRS	0.02	0.06	3.9

Conclusion

In this article, the possible approaches to solve the *data splitting* problem were discussed, which is an important part of cross-validation when training or comparing computational models (namely simple random sampling, deterministic methods, DUPLEX, stratified sampling and others). In most of the applications, the simple random sampling is used. However, for various types of datasets, other sophisticated statistical sampling techniques could be adopted to reduce the variance of the model performance.

Experimental results obtained so far confirm that for simple low-dimensional datasets, any of the methods can be used to achieve similar results – in such cases, the simple and efficient methods, such as SRS, may be prefered. When dealing with complex high-dimensional datasets, the SRS often leads to higher bias and variance of the model error than the more sophisticated methods such as the stratified random sampling and DUPLEX. However, the stratified random sampling significantly exceeds DUPLEX in time complexity and therefore seems to be more suitable for the large and complex datasets. Unfortunately, this method is very sensitive to the choice of its parameters.

For this reason, our further research will be focused on investigating more stable variants of the stratified random sampling. The experiments (performed so far just for the hold out cross-validation) may also be repeated also for the case of k-fold cross-validation.

Acknowledgments. The author would like to thank her advisor Doc. RNDr. Iveta Mrázová, CSc. for her useful advices and overall help. This work was supported by Grant Agency of the Czech Republic under the Grant-No. 201/09/H057 and by Grant Agency of Charles University under the Grant-No. 17608.

References

Baxter, C., Stanley, S., Zhang, Q., and Smith, D., Developing artificial neural network process models: a guide for drinking water utilities., *6th Environmental Engineering Specialty Conf. of the CSCE*, *11*, 376–383, 2000.

Bowden, G. J., Maier, H. R., and Dandy, G. C., Optimal division of data for neural network models in water resources applications, *Water Resource Research*, *38*, 1–11, 2002.

Bowden, G. J., Dandy, G. C., and Maier, H. R., Input determination for neural network models in water resources applications. part 1background and methodology, *Journal of Hydrology*, *301*, 75–92, 2005.

Cochran, W. G., *Sampling Techniques, 3rd Edition*, John Wiley, 1977.

Fernandes, S., Kamienski, C., Kelner, J., Mariz, D., and Sadok, D., A stratified traffic sampling methodology for seeing the big picture, *Comput. Netw.*, *52*, 2677–2689, 2008.

Kaufman, L. and Rousseeuw, P. J., *Finding Groups in Data An Introduction to Cluster Analysis*, John Wiley, New York, 1990.

Kennard, R. and Stone, L., Computer aided design of experiments, *Technometrics*, *11*, 137–148, 1969.

Kingston, G., Bayesian artificial neural networks in water resources engineering., Ph.d., The University of Adelaide., 2006.

Kononenko, I. and Kukar, M., *Machine Learning and Data Mining: Introduction to Principles and Algorithms*, Horwood Publishing Limited, 2007.

Lohr, S. L., *Sampling: Design and Analysis*, Duxbury Press, 1 edn., 1999.

May, R. J., Maier, H. R., and Dandy, G. C., Data splitting for artificial neural networks using som-based stratified sampling, *Neural Networks*, *23*, 283–94, 2010.

Mitchell, T. M., *Machine Learning*, McGraw-Hill, New York, 1997.

Oja, E., Unsupervised learning in neural computation, *Natural computing*, *287*, 187–207, 2002.

Picard, R. R. and Cook, R. D., Cross-validation of regression models, *Journal of the American Statistical Association*, *79*, 575–583, 1984.

Reed, R. D. and Marks, R. J., *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, Cambridge, MA, USA, 1998.

Refaeilzadeh, P., Tang, L., and Liu, H., Cross-validation, in *Encyclopedia of Database Systems*, pp. 532–538, Springer US, 2009.

Snee, R. D., Computer aided design of experiments, *Technometrics*, *19*, 415–428, 1977.

Zhang, G. P. and Berardi, V. L., Time series forecasting with neural network ensembles: An application for exchange rate prediction, *The Journal of the Operational Research Society*, *52*, 652–664, 2001.