Data Partitioning Evaluation Measures for Classifier Ensembles

Rozita A. Dara, Masoud Makrehchi, and Mohamed S. Kamel

Pattern Analysis and Machine Intelligence Laboratory,
University of Waterloo, Waterloo, Ont., Canada, N2L-3G1
{rdara, makrechi, mkamel}@pami.uwaterloo.ca

Abstract. Training data modification has shown to be a successful technique for the design of classifier ensemble. Current study is concerned with the analysis of different types of training set distribution and their impact on the generalization capability of multiple classifier systems. To provide a comparative study, several probabilistic measures have been proposed to assess data partitions with different characteristics and distributions. Based on these measures, a large number of disjoint training partitions were generated and used to construct classifier ensembles. Empirical assessment of the resulted ensembles and their performances have provided insights into the selection of appropriate evaluation measures as well as construction of efficient population of partitions.

1 Introduction

In the traditional pattern classification design strategies, experimental assessment of the performance of several classifiers, with the aim of selecting the best classifier, used to be considered. In a diversion from the traditional approach, the use of classifier ensemble has been proposed as an alternative technique to improve classification accuracy [1, 2]. The design of classifier ensemble can take place at four levels of data, feature, classifier and aggregation [3]. Most of the work in MCS has focused on developing new aggregation methods. In addition, the base classifiers and the architecture of the system have been the center of attention [4]. An alternative design methodology is based on training base classifiers using overlapped or disjoint feature subsets [5]. Moreover, another popular ensemble technique is accomplished by modifying input data (e.g. [6],[7]). There has been a lot of emphasis on this method, since it promotes diversity among individual classifiers [3].

The focus of this paper is on data level techniques. Data level partitioning strategies can be discussed from different perspectives. For example, from sharing point of view [8]. Majority of the methods use highly overlapped (shared) partitions to train the classifiers, while few others [9] apply disjoint subsets of the data. The latter approach is mostly used when dealing with large datasets. Another aspect of data level combining techniques is the type of information applied in the partitioning stage. Some methods, such as boosting and feature-

N.C. Oza et al. (Eds.): MCS 2005, LNCS 3541, pp. 306–315, 2005.

based, use sort of a knowledge provided by the output of the classifiers to optimize the partitions. A key issue in this method is the computational effort required. Alternatively, some other techniques are based on random selection of data patterns, such as Bagging. Despite the significant number of theoretical and empirical studies on data partitioning methods, no criteria or guidelines for the use of different methods exist.

The present work introduces a novel direction in data level combining strategies. We suggest that a thorough understanding of the advantages and disadvantages of different data modification techniques will lead to more intelligent choices when designing MCS architectures. To achieve this goal, we first categorized data level combining techniques from a new perspective. Then, we proposed several correlation measures for training partitions, which will be discussed in the following sections. These measures were applied to estimate the degree and type of information provided by each partition. Through the splitting of the original training data using different class distributions and distances, we evaluated and compared the impact of different training partitions on the system performance. The main focus of this study was on the methods in which partitioning takes place independent of classifiers, filter approach, without a pass through training stage (Section 2). The empirical assessment of these measures has provided some insights into how training patterns are utilized in MCS system.

2 Categorization of Data Level Combining Techniques

Blum and Langley [10] have identified three types of feature selection methods: Embedded, Filter and Wrapper. We adopt the same terminologies for data and feature space partitioning in MCS. In this section, partitioning techniques are categorized into two groups: wrapper and filter approaches.

In the wrapper method, subsets are evaluated and partitioned based on the output of the base classifiers before making the final decision. Boosting and feature-based are methods that fall into this category. Boosting, proposed by Freund and Schapire, [6] is a popular method in which classifiers are built in sequence. At each step of boosting, training data for the next classifier is selected by assigning weights to data patterns that have been misclassified by the previous classifier. Kamel and Wanas [11] proposed an *evolving* training algorithm where base classifiers utilize the outcome of an aggregation method to rearrange the training subsets. Hierarchical Mixtures of Experts [12], and Learn++ [13] fit into the category of the wrapper approach, since partitioning depends on the output of the base classifiers. The wrapper approach has three drawbacks: *(i)* it is time demanding and expensive since the population of partitions has to be evaluated and optimized step by step before finalizing the training, *(ii)* decisions on base classifier and the MCS architecture have to be made in advance, and *(iii)* a powerful architecture and/or base classifiers may be traded off by the weakness of selected partitions.

In the filter method, however, subsets are partitioned before training. Bagging [7] is a well-recognized combining technique that partitions training data

before constructing the base classifiers. Bagging presents each classifier with a set of data points sampled uniformly with replacement from the original data. The final classification decision is attained by taking the majority vote of the generated ensemble. K fold-corssvalidation is another example of filter approach. In this method, the training set is randomly divided into k subsets. Leaving one subset out, each classifier is presented with k-1 subsets for training. A more deterministic filter approach has been proposed in [14], where original training data is partitioned using clustering techniques. Each cluster is then used to train a new classifier. Filter method is less expensive and selection of the architecture and base classifiers does not have to be made in advance. However, the difficulty in this method is defining a cost function, which can estimate efficiency and capability of the generated partitions.

3 Data Partitioning Evaluation Measures

We developed and implemented several measures to evaluate partitions and their effects on MCS performance. These measures can be grouped into two main categories of feature-based and non-feature-based. In the non-feature-based measures, information provided by the feature space is not considered and only the class labels of training data are taken into account. While in feature-based methods, all measures are calculated based on the feature information. Non-feature-based measures have been previously discussed in [15]. The main focus of this paper is on feature-based measures. In addition to the mentioned categorization, we can apply the proposed measures at three levels of class, individual partition, and overall collection of partitions. We developed distance measures that utilized patterns from one class, *intraclass* distance, more than one class, *interclass* distance, one individual partition, *intra-partition* distance, and more than one partition, *inter-partition* distance. These categorizations are summarized in Table 1.

Table 1. Four Different Conditions for Location of Data Patterns

	same partitions	**different partitions**
same classes	intra-class intra-partition	intra-class inter-partition
different classes	inter-class intra-partition	inter-class inter-partition

3.1 Feature-Based Measures

Measure of distance can be used as a criterion to evaluate data partitioning schemes. Distance of the data patterns can be interpreted as *density*, within one class, or *correlation* between two classes or more. To calculate the distance of two sample points, the use of feature space of the data is inevitable. Therefore, all proposed measures in this section are addressed as feature-based. We employed Euclidean, Bhattacharyya, and Mahalanobis distance measures.

Euclidean Distance: The most popular and simple distance measure is Euclidean distance [16]. Defining a metric space, Euclidean measure is denoted as:

$$d_E = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are feature vectors selected from spcific data populations. Each vector has n features, representing the dimension of the feature space.

Mahalanobis Distance: One drawback of Euclidean distance is that it does not consider the correlation between two features. Mahalanobis distance, on the other hand, takes into account the correlation of the features. The Mahalanobis distance between two feature vectors $\mathbf{x}$ and $\mathbf{y}$ are represented as:

$$d_M = \sqrt{(\mathbf{x}-\mathbf{y})^T \Sigma^{-1}(\mathbf{x}-\mathbf{y})} \tag{2}$$

where Σ is the within-group (pooled) covariance matrix of two samples. Σ is defined as

$$\Sigma = \frac{(n_x-1)\Sigma_x + (n_y-1)\Sigma_y}{n_x+n_y+2} \tag{3}$$

where, n_x and n_y are the size of x and y samples, Σ_x and Σ_y are the covariance matrices of x and y accordingly.

Bhattacharyya Distance: Similar to Mahalanobis, Bhattacharyya distance measure is widely used as a measure of class separability, because of its precision and relation with the Bayes error [16].

$$d_B = \frac{1}{8}\sqrt{(x-y)^T(\frac{\Sigma_x+\Sigma_y}{2})^{-1}(x-y)} + \frac{1}{2}ln(\frac{|(\Sigma_x+\Sigma_y)/2|}{\sqrt{|\Sigma_x|.|\Sigma_y|}}). \tag{4}$$

Applying these distance measures, we can draw an overall evaluation value for a set of samples or partitions by averaging all pairwise distances in that population. Each population can belong to one of the following structures; *(i)* C_j^i, the j^{th} class in the i^{th} partition, *(ii)* P_i, the i^{th} partition which consists of c classes, *(iii)* C_j, the j^{th} class in the collection which is broken into m partitions, and *(iv)* the whole collection divided into m partitions. To calculate correlation of each of the mentioned structures, there are two options. We can either consider all patterns in the selected population and their distances from each other, or consider only center of that population. In the first approach, if a population consists of N samples, we need to calculate $\frac{N(N-1)}{2}$ distances. The second approach is much simpler, since instead of using pairwise measures, the distances of all samples from their own population center, or another population center, are calculated. This simplification reduces the number of operations to N distance measures.

The following notations are used through out the paper: μ_j^i or ν_j^i represent the mean (center) of the j^{th} class in the i^{th} partition and μ_j or ν_j the mean of the j^{th} class. μ^i or ν^i is the mean of the i^{th} partition, $x_{k,j}^i$ is considered as the feature vector of the k^{th} sample of j^{th} class in the i^{th} partition, and $d(x,y)$ represent

the distance between x and y. In addition, D_j^i is the correlation among the j^{th} class in partition i, D_j is the correlation among class j, and finally, D^i is the correlation of i^{th} partition. The number of partitions and classes are represented by m and c. n_{ij} is the number of samples in the j^{th} class in i^{th} partition, M_i is the number of samples in partition i and N_j is the size of class j.

Intra-class, Intra-partition Distance: The population, in this category, consists of patterns from a specific type of class. Correlation is calculated as

$$D = \sum_{i=1}^{m} \sum_{j=1}^{c} p_{c_j} \sum_{k=1}^{n_i j} d(x_{k,j}^i, \mu_j^i), \quad p_{c_j} = \frac{n_{ij}}{M_i}. \tag{5}$$

Inter-class, Intra-partition Distance: Here, correlation among two or more classes in a partition is considered. We can either consider the actual patterns or their means. Their distances from the overall mean of partition can be considered as well.

$$D = \sum_{i=1}^{m} \sum_{j=1}^{c} p_{c_j} d(\mu^i, \mu_j^i), \quad p_{c_j} = \frac{n_{ij}}{M_i}. \tag{6}$$

Intra-class, Inter-partition Distance: Correlation of every class, which is broken down into m partitions, is the basis for evaluation. It is important to note that instead of the mean, μ, actual patterns can be used as well.

$$D = \sum_{j=1}^{c} \sum_{i=1}^{m} p_{p_i} d(\mu_j, \mu_j^i), \quad p_{p_i} = \frac{n_{ij}}{N_j}. \tag{7}$$

Inter-class, Inter-partition Distance: The overall correlation of a pool of partitions is calculated by considering the distances of *i)* centers of all classes in the partition, or *ii)* centers of all partitions from the overall mean.

$$D = \sum_{i=1}^{m} \sum_{j=1}^{c} d(\nu_j^i, \mu_j^i) \tag{8}$$

4 Experimental Setup

We carried out a set of experiments in order to assess the proposed evaluation measures and the impact of large set of partitions, generated with different distributions and distances, on generalization ability of the MCS. We considered all the previously discussed categories, including all three distance measure in four categories, either pairwise or non-pairwise, population center or actual patterns themselves, as well as all types of inter/intra class/partition measures. This section highlights a summary of the important observations.

We present the results for two of the benchmark datasets; both of them are available within the UCI Machine Learning Repository [18]. The first dataset is Pima Indians Diabetes, which contains 8 attributes and 2 classes. It includes 768

patterns with no missing values. The second one is again a real-world dataset, satimage, which contains 36 attributes, 6 classes and 6435 instances. Linear discriminant classifiers were used in the experiments. The use of stable classifiers was dictated by the need of obtaining consistent results by eliminating drastic fluctuation in the system performance caused by classifiers themselves. For the results presented in this paper, number of classifiers were set to 7 for diabetes and 9 for Satimage data. These selection of the ensemble sizes was based on the size and complexity of the data. The use of disjoint training subsets prohibited us from the use of larger ensembles. The product of the estimated posterior probabilities by the base classifiers was used as combining rule.

To examine our proposed measures, we considered disjoint subsets for training. Training subsets were obtained in a deterministic way, by opposing and controlling *bias* among partitions. Bias is defined as a fraction of a training data (%) that belongs to a specific group. In these experiments, bias was obtained by grouping training data based on the instances distances from the center of classes or training data itself. The initial set of partitions were obtained through opposing 100% bias to all partitions. Partitions with 100% bias was obtained by this procedure. We first sorted out the patterns based on their distance from center of the class or training data. We distributed the sorted patterns in such a way that first partition contained the largest distanced patterns, the second partition contained the second largest distanced patterns, and so on. It is important to note that the size of partitions has remained balanced through all the stages, as well as the type of class diversity. This set of partitions has remained the baseline for producing next sets of partitions by gradually reducing bias from 100% to 0% (Algo. 1).

Algorithm 1 Data Partitioning Evaluation Scheme

$b \leftarrow 10$
for $1 \leq b$,
 divide the data into training and test subsets, proportionally half and half
 determin the number of partitions, m
 initialize the first training partitions by forcing 100% bias (discussed above), p_0
 $bias \leftarrow 95\%$
 while *"bias is non-zero"*
  randomly select $(100 - bias)\%$ of the data, T_b
  construct new partitions based on the initial set of partitions p_0
  and distribute T_b among these subsets in such a way that there is no overlap
  $bias \leftarrow bias - step$; *step* is a predefined number
 end while
 calculate the correlation of all population of partitions using the measures discussed above train individual classifiers by partitioned training data and combine their outputs
end for

As discussed, we applied Euclidean, Mahalanobis, and Bhattacharyya distance measures to estimate correlation among training sets. Among all these

measures, Euclidean distance has shown to have the largest inconsistency and fluctuations. The other two were more stable and demonstrated similar trends. Since feature-vectors do not have the same variance in each dimension, it is necessary to use a normalized distance measure. Mahalanobis, and Bhattacharyya have this property, as well as the fact that they are a good approximation of Bayes error.

The test error rates, y-axis, against correlation measures, x-axis, are summarized in Figures 1-3, for Pima and Satimage datasets. The presented results have been averaged over 10 different population sets. In addition, we evaluated the performance of generated partitions against bagging; presented as a straight line in all figures. Bagging was trained and tested with 20 classifiers and the same training and test sets that were used for the rest of the experiments. The average of 10 runs has been reported for bagging as well.

The effect of intra-class distance in each partition is depicted in Figures 1(a) and (b). As it is shown, the smaller the distance of instances get from their

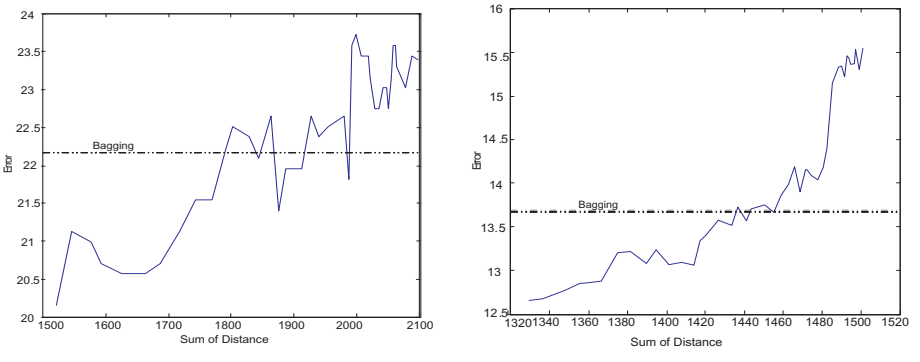

Fig. 1. Intra-class, Intra-partition Measure: a)Pima b)Satimage

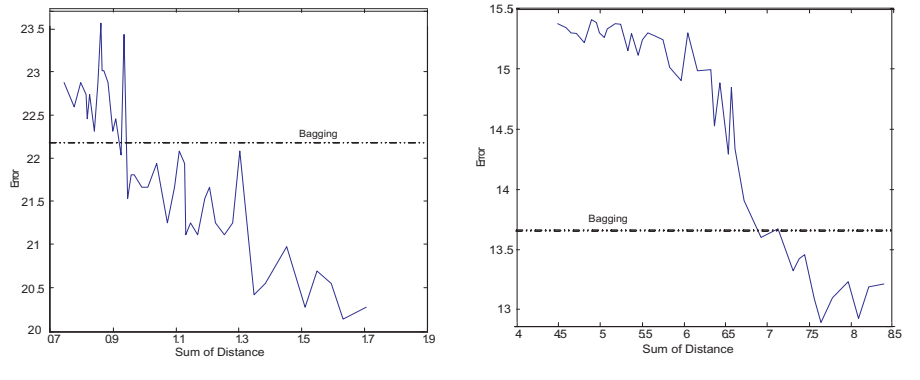

Fig. 2. Inter-class, Intra-partition Measure: a)Pima b)Satimage

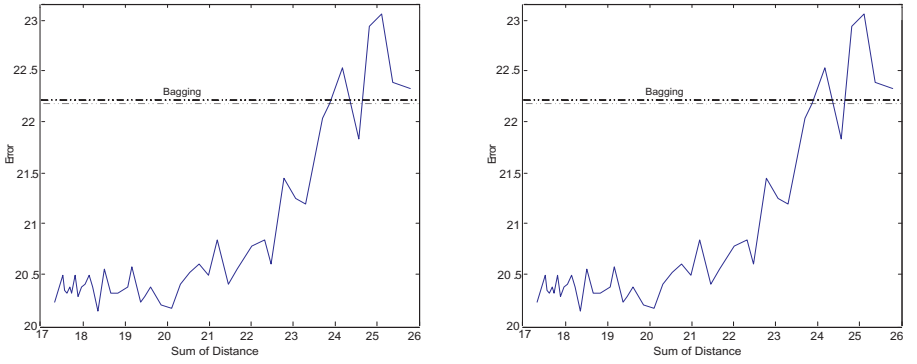

Fig. 3. a)Intra-class, Inter-partition Measure (Pima), b)Using Training Data Mean for Partitioning (Satimage)

mean (larger density), the more improvement can be seen in the accuracy. Similar investigation has been carried out among partitions, inter-partition measure. This time the mean of overall class in the training data has been considered, instead of the mean of the class in the partition (Fig. 3(a)). The error tends to increase with the increase of sum of the distances. This behaviour was expected in both cases, as it is well known in classical pattern recognition that larger class density increases the generalization ability. We also considered inter-class, intra-partition distance measures (Fig. 2 (a) and (b)). The overall trend in this case study was that larger inter-class distance resulted in lower error. This is due to the fact that by larger inter-class distance, the likelihood of having overlaps among classes decreases, which makes it easier for the system to distinguish different classes. As discussed before, patterns were forced to distribute either based on their distances from the mean of class or training data. By using the mean

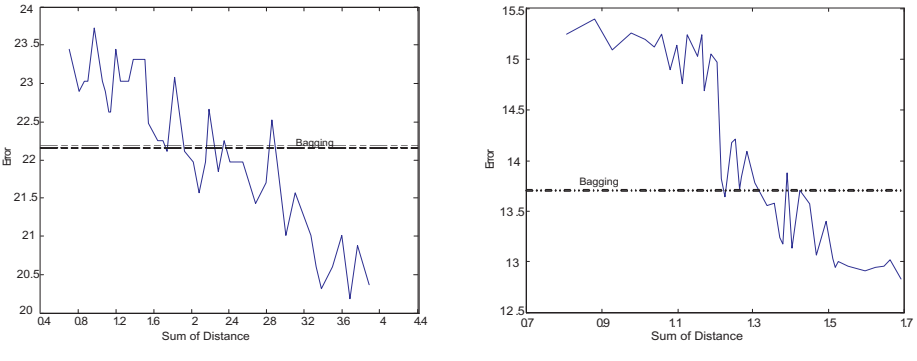

Fig. 4. Inter-partition Measure: a)Pima b) Satimage

of the training data (Fig. 3(a)), the pattern of change was the same, however, the range of difference between worst and best performance using class mean is larger, 2.6% versus 1.5%. Considering inter-partition feature-based measures (overall collection), we did not know what to expect when examining the distance between mean of training subsets and the mean of original training data. Results suggest that larger distances among the means improve MCS classification capability (Fig. 4(a) and (b)).

As illustrated in the figures, some of the generated disjoint partitions outperform bagging up to 2%. An important observation was that by examining each partition set individually, their distances, and their corresponding ensemble error, it became clear that initial partition set (generated with 100% bias) had the lowest error. This suggest that the larger the bias gets, the better generalization ability multiple classifiers obtain. More detailed investigation needs to be carried out to be able to make more reliable conclusions. However, the current findings promise to offer new design criteria for filter type partitioning strategy using feature-based measures.

5 Conclusion

In this paper, we proposed several feature-based distance measures, with which correlation among training partitions can be estimated. Using overall distance measures in different scopes of classes and partitions, large number of partitions were generated, and their impacts on classification accuracy were evaluated. It is important to note that we were not aiming at finding the best classification results. We were mostly searching for a linkage between data partitioning strategies, and performance of MCS. As a result, we used the simplest possible classification and aggregation algorithms to design ensemble members. The use of more sophisticated methods will be the focus of our future work.

The above observations support the notion that the proposed evaluation measures are beneficial in obtaining a nearly optimal partitioning solution, considering that training data partitioning based on these measures is less expensive than clustering and many other existing techniques. In addition, the results show that MCS error is a non-linear function of overall distance measure(s). Therefore, it is expected to be able to find some optimal or sub-optimal points in which MCS performance is at its best. These observations address an optimization procedure which is our future work. In addition, findings of this study can be used to optimize existing combining architectures and techniques.

References

1. J. Kittler and F. Roli, (Eds.), "Multiple Classifiers Systems", LNCS 3077, 5th Int. Workshop on MCS, Cagliari, Italy, 2004.
2. J. Kittler, and F. Roli, (Eds.), "Multiple Classifiers Systems", LNCS 2709, 4th Int. Workshop on MCS, Gambridge, UK, 2003.

3. L. Kuncheva, "Combining Pattern Classifiers: Methods and Algorithms", Wiley, 2004.
4. A. Sharkey, "Types of Multinet Systems," Third Int. Workshop on MCS, LNCS 2364 , Cagliari, Italy, , (2002) 108–117.
5. K. Tumer, and N. Oza, "Input Decimated Ensembles," Pattern Analysis and Applications, **6(1)**, (2003) 65-77.
6. Y. Freund, and R. Schapire, "Experiments with a New Boosting Algorithm," Proc. of the 13th Int. Conf. on Machine Learning, Bari Italy (1996) 148–156.
7. L. Breiman, Bagging Predictors. Machine Learning, **24(2)** (1996) 123–140.
8. R. Dara, and M. Kamel, "Effect of Sharing Training Patterns on the Performance of Classifier Ensemble," Proc. of Int. IEEE Conf. on System Man Cybernetics, The Hague, The Netherlands, (2004) 1220–1225.
9. N. Chawla, T. Moore, L. Hall, L. Bowyer, P. Kegelmeyer, and C. Springer, "Distributed Learning with Bagging-like Performance" Pattern Recognition Letters, 24 (2003) 455–471.
10. A. Blum, and P. Langley, "Selection of Relevant Features and Examples in Machine Learning", Artificial Intellignce, **97(1-2)**, (1997) 245–271.
11. M. Kamel, and N. Wanas, Data Dependence in Combining Classifiers. Fourth Int. Workshop on MCS, Guilford UK (2003) 1–14.
12. W. Jiang, and M. Tanner Hierarchical Mixtures of Experts for Generalized Linear Models. Neural Computation **11** (1999) 1183–1198.
13. D. Parikh, M. Kim, J. Oagaro , S. Mandayam and R. Polikar, "Combining classifiers for multisensor data fusion," Proc. of Int. IEEE Conf. on System Man Cybernetics, The Hague, The Netherlands, (2004), 1232–1237.
14. D. Frosyniotis, A. Stafylopatis, and A. Likas, "A Divide-and-Conquer Method for Mutli-Net Classifiers," Pattern Analysis and Applications, **6**, (2003) 32-40.
15. R. Dara, M. Makrehchi, and M. Kamel, " An Information-Theoretic Measure to Evaluate Data Partitions in Multiple Classifiers" Proc. of Int. IEEE Conf. on System Man Cybernetics, The Hague, The Netherlands, (2004) 4826–4831.
16. K. Fukunaga "Introduction to Statistical Pattern Recognition," Oxford University Press, 1990.
17. R. Duda, P. Hart, and D. Strok, "Pattern Recognition," John Wiley and Sons, 2000.
18. D. Blake, and C. Merz, UCI Repository of machine learning databases, [http://www.ics.uci.edu/ mlearn/MLRepository.html].