Dynamic Integration of Data Mining Methods Using Selection in a Knowledge Discovery Management System

Seppo PUURONEN, Vagan TERZIYAN, Alexey TSYMBAL
University of Jyväskylä, P.O.Box 35, FIN-40351 Jyväskylä, Finland

Abstract. One of the important directions in improvement of the data-mining and knowledge discovery methods is the integration of multiple classification techniques. An integration technique should estimate and then select the most appropriate component classifiers from an ensemble of classifiers. We discuss an advanced dynamic integration technique with multiple classifiers as one variation of the stacked generalization method based on the assumption that each component classifier is the best inside some sub areas of the application domain. In the learning phase a performance matrix of each component classifier is derived and it is then used during the application phase to estimate the performance of each component classifier with new instances.

1. Introduction

Data mining is the process of finding previously unknown and potentially interesting patterns and relations in large databases [3]. Numerous data mining methods have recently been developed to extract knowledge from large databases. In many cases it is necessary to evaluate and then select the most appropriate data-mining method or a group of methods. Often the method selection is done statically without analyzing each new instance. When the method selection is done dynamically taking into account the characteristics of each instance better results are achieved usually.

A general architecture of a knowledge discovery management system (KDMS) is presented in Figure 1. The data base management subsystem performs data storage, retrieval, and manipulation operations. It is important that this subsystem supports distributed and heterogeneous data and complex data types, adopts a client/server architecture, and can access databases of standard wide-used formats. The data preprocessing subsystem provides tools for data preparation, for data mining and exploratory data analysis before the use of data mining techniques. These tools are used to lower the level of noise, handle missing data fields, reduce data and make data projections. The data mining subsystem incorporates different mining techniques for creating models and extracting patterns of data. It is important that this subsystem is open allowing easy addition of new techniques when these are available. The method selection subsystem helps a user to select an appropriate data mining method.

The result visualization subsystem includes tools for visualization of models being built and patterns discovered. This subsystem helps a user to interpret and evaluate extracted patterns and models being essential for the process of obtaining new knowledge. The user interface guides the user through the discovery process helping him to manage the power and complexity of knowledge discovery.

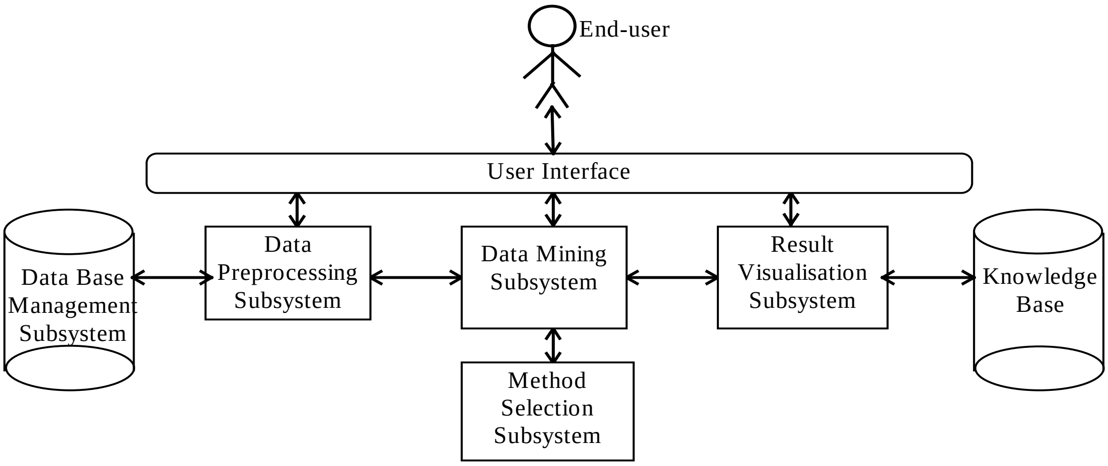

Figure 1: An architecture of a KDMS

In a variety of applications, researchers try to combine efforts to learn to create and combine an ensemble of classifiers. For example, in [2] integrating multiple classifiers has been shown to be one of the four most important directions in machine learning research. The main discovery is that ensembles are often much more accurate than the individual classifiers. In [11] the two advantages of combining classifiers were shown: (1) the possibility that by combining a set of classifiers, we may be able to perform classification better than with any classifier alone, and (2) the accuracy of a sophisticated classifier may be increased by combining its predictions with those made by an unsophisticated classifier.

We base on the assumption that each classifier is the best inside some sub domains of the application domain. In this paper, we develop a dynamic integration technique which tries to estimate and benefit from these competence areas of the classifiers. In chapter 2 we discuss published related work about integrating multiple classifiers. In chapter 3 our dynamic integration method is presented and we conclude in chapter 4.

2. Integrating multiple classifiers

In this chapter we shortly review related work published about integrating multiple classifiers. Integrating multiple classifiers to improve classification results has been an area of research in machine learning and neural networks. Different approaches to integrate multiple classifiers and their applications have been considered for example in [1, 2, 4, 5-7, 8, 10-15]. The challenge of the integration problem is to decide which classifiers to rely on or how to combine results of several classifiers.

The integration problem can be defined as follows. Let us suppose that a training set **T** and an ensemble of classifiers **C** are given. Let the training set **T** be: $\{(\mathbf{x}_i, y_i), i = 1,\dots,n\}$, where n is the number of the training instances, $\mathbf{x}_i$ is the vector of the attributes $\{x_j\}, j = 1,\dots,l$ of the i-th training instance (the values of the attributes can be numeric, nominal, or symbolic), and $y_i \in \{c_1,\dots,c_k\}$ is the actual class of the i-th instance (k is the number of classes). Let the ensemble **C** of classifiers be: $\{C_1,\dots,C_m\}$, where m is the number of the available classifiers (we will call them *component classifiers*). Each component classifier is either derived using some learning algorithm or hand-crafted using some heuristic knowledge. Let a new instance $\mathbf{x}^*$ be an assignment of values to the vector of the attributes $\{x_j\}$ without known actual classification.

The integration problem is to use the ensemble $\mathbf{C}$ of the classifiers to classify the new instance $\mathbf{x}^*$ as accurately as possible. Recently two basic approaches are used to integrate multiple component classifiers of an ensemble. In the first approach, all the component classifiers produce their classification results which are then *combined.* In the second approach the best classifier of the ensembled ones is first *selected* and then it is used to produce the classification result.

2.1. The combining approach

Several effective methods have been proposed to combine the results of the ensembled component classifiers. In the simple voting method (also called majority voting and Select All Majority, SAM) [6] the prediction of each component classifier is an equally weighted vote in the result favor produced by the component classifier. The result which receives most votes is selected. There are also more sophisticated methods to combine the results of the classifiers. These include the stacking (stacked generalization) architecture [15], the SCANN method based on the correspondence analysis and the nearest neighbor procedure [7], and combining minimal nearest neighbor classifiers within the stacked generalization framework [11]. Different versions of resampling (boosting, bagging, and cross-validated resampling) that use a learning algorithm to train the component classifiers using sub samples of the training set and then simple voting to combine their results have also been suggested for example in [2,10].

Two effective classifiers' combining strategies based on hierarchical stacked generalization (called an arbiter and a combiner) were analyzed and experimented in [1]. It was shown that the hierarchical combination approach was able to sustain the same level of accuracy as a global classifier trained using the entire data set. Even when the component classifiers were trained using only subsets of the entire data set. The stacked generalization model considered by Wolpert [15] is one of the most widely studied and used today. He has presented a basic scheme of the stacked generalization architecture discussed below.

The basic scheme of the stacked generalization architecture consists of an ensemble of component classifiers $\mathbf{C} = \{C_1, \ldots, C_m\}$ that form the first layer, and of a single combining classifier ***M*** of the second layer of the hierarchical structure. When a new instance $\mathbf{x}^*$ is classified, then first, each of the component classifiers C_i is launched to give its prediction of the instance's class $C_i(\mathbf{x}^*)$. Second, the combining classifier ***M*** uses the predictions of the component classifiers to derive the final prediction of the whole composite classifier $\boldsymbol{M}(C_1(\mathbf{x}^*), \ldots, C_m(\mathbf{x}^*))$. The included classifiers are created in two phases: first, the component classifiers C_i are created and second, the combining classifier is trained using meta-level training set which is composed of the predictions of the component classifiers $\{C_i(\mathbf{x}_j)\}$ and the training set $\mathbf{T}$ [11,15].

The stacked generalization architecture for classifier combination has still many open questions. For example, there are currently no strict rules saying which component classifiers should be used and what features of the training set should be used to train the combining classifier. Different combining algorithms have been considered for example, the classic boosting using simple voting [10], Skalak [11] discusses ID3 for combining nearest neighbor classifiers, and in [7] the nearest neighbor classification is proposed to search the space of correspondence analysis results.

2.2. Selection approach

Several effective approaches have recently been proposed also for *selection* of the best classifier. In CVM (Cross-Validation Majority) [6] the cross-validation accuracy for each classifier is estimated with the training data, and a classifier with the highest accuracy is selected. More sophisticated selection approaches include, for example, estimating the local accuracy of component classifiers by considering errors made by component classifiers in instances with the same response pattern [6] and learning meta-level classifiers ("referees") that predict whether a component classifier is giving right or wrong prediction for a new instance [8]. In [12-14] we have considered an application of classifier selection to medical diagnostics. It predicts the local accuracy of component classifiers by analyzing the accuracy in neighboring learning instances.

Classifier selection methods can be divided into two groups: *static* and *dynamic* selection methods. The static methods propose one "best" classifier for the whole data space, while the dynamic methods takes into account the characteristics of each new instance individually. For example CVM approach belongs to the group of static selection methods, while the other selection methods above and the one proposed in this paper are dynamic.

3. Dynamic integration of classifiers

In this chapter we consider a new variation of stacked generalization which uses a metric to estimate the errors of component classifiers locally. Rather than trying to train a meta-level classifier that will predict a class using predictions of component classifiers as in stacked generalization, we propose to train a meta-level classifier that is able to estimate the errors of the component classifiers for each new input instance. These estimates are used to select the component classifier dynamically. The goal is to use each component classifier just in the sub domain for which it is the most reliable one, and thus to achieve overall results that can be better than those achieved using the best individual classifier globally. We first describe a space model and then present our dynamic approach.

3.1. Space model

Solutions of the classification problem using learning are commonly based on the assumption that the distribution of the instances of a training set is same as the distribution of domain instances including null-entropy areas, i.e. instances of each class concentrated in some sub areas of the domain space. Learning algorithm creates a space model of the distribution using learning set. The space model does not describe the space exactly. In fact, the accuracy of a model depends on many factors as the size and shape of decision boundaries, the number of null-entropy areas (problem-depended factors), the completeness and cleanness of the training set (sample-depended factors) and the individual peculiarities of the learning algorithm (algorithm-depended factors). Figure 2 represents a simple example with the binary classification rule: class is “Yes” if $x_2 > x_1$ and “No” otherwise, and a classifier being built by the C4.5 algorithm [9] that splits the space into rectangles, each of which corresponds to a leaf of the decision tree. It can be seen that the points misclassified by the model (black areas in Figure 2) are not uniformly distributed thorough the space.

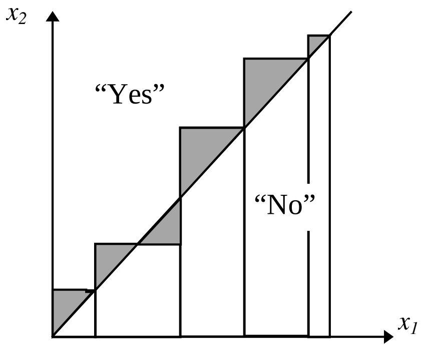

Figure 2. An example of a space model built by C4.5 [9]

The entire space can be considered as a set of null-entropy areas of instances with categories “a classifier makes correct classification“ and “a classifier makes incorrect classification“. Our dynamic approach attempts to create a meta-model describing this division of the space. This meta-model is then used to predict the errors of the component classifiers in handling new instances.

3.2. Dynamic approach

Our dynamic approach has characteristics of eager machine learning because it collects during the learning phase information needed later when new instances are classified. It also has characteristics of lazy machine learning because it does not fix the hypothesis during the learning phase being able to take into account the characteristics of new instances when they are classified.

The approach contains two phases. In the first, *learning phase*, a matrix describing the classifiers' performances for each training instance is formed and stacked. In the second, *application phase*, the combining classifier is used to predict the performance of each component classifier for a new instance.

Our approach can be considered as a particular case of the stacked generalization architecture. Instead of the component classifier predictions, we use information about the performance of component classifiers for each training instance. In the learning phase, the training set **T** is partitioned into folds, the errors of the component classifiers for each instance of the training set and for each of the m component classifiers E_j are collected into n vectors. The values of these vectors can be binary (i.e. classifier gives correct or incorrect result) or can represent corresponding misclassification costs. This information about the performance of component classifiers is then stacked and it is further used together with the initial training set as meta-level knowledge for estimating errors in new instances.

In the application phase two approaches are used. In the dynamic selection (DS) approach the classification error E_j^* is predicted for each component classifier C_j using the WNN procedure and a classifier with the smallest error estimate is selected to the final classification. In the dynamic voting (DV) approach each component classifier C_j has a weight W_j that depends on the classifier's local performance and the final classification is achieved by weighted voting.

4. Conclusion

In data mining one of the key problems is selection of the most appropriate data-mining method. Our goal was to further enhance dynamic integration of multiple classifiers. We represented a meta-classification framework. It consists of two levels: the component classifier level and the combining classifier. Assuming that each component classifier is best inside some sub domains of the application domain we proposed a variation of the stacked generalization method where classifiers of different nature can be integrated. The performance matrix for component classifiers is derived during the learning phase. It is then used in the application phase to estimate the performance of the classifiers. The final selection is made using either dynamic selection or dynamic voting. Evaluations with benchmark data have been promising but further research with real data are still needed to evaluate the practical value of our approach.

Acknowledgement: This research is partly supported by the Academy of Finland.

References

[1] P. Chan and S. Stolfo, On the Accuracy of Meta-Learning for Scalable Data Mining. *Intelligent Information Systems* **8** (1997) 5-28.

[2] T. Dietterich, Machine Learning Research: Four Current Directions. *AI Magazine* **18** (1997) 97-136.

[3] U. Fayyad *et al.*, Advances in Knowledge Discovery and Data Mining. AAAI/ MIT Press , 1997.

[4] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Proceedings of IJCAI'95, 1995.

[5] M. Koppel and S. Engelson, Integrating Multiple Classifiers by Finding their Areas of Expertise. In: AAAI-96 Workshop On Integrating Multiple Learning Models, 1996, pp. 53-58.

[6] C. Merz, Dynamical Selection of Learning Algorithms. In: D. Fisher, H.-J.Lenz (Eds.), Learning from Data, Artificial Intelligence and Statistics, Springer Verlag, NY, 1996.

[7] C. Merz, Combining Classifiers Using Correspondence Analysis. In: Advances in Neural Information Processing Systems (NIPS 10), 1998, to appear.

[8] J. Ortega *at al.*, Arbitrating Among Competing Classifiers Using Learned Referees, *Machine Learning*, 1998, to appear.

[9] J. Quinlan, C4.5 Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[10] R. Schapire, Using Output Codes to Boost Multiclass Learning Problems. In: Machine Learning: Proceedings of the Fourteenth International Conference, 1997, pp. 313-321.

[11] D. Skalak, Combining Nearest Neighbor Classifiers. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts, Amherst, MA, 1997.

[12] V. Terziyan, A. Tsymbal and S. Puuronen, The Decision Support System for Telemedicine Based on Multiple Expertise. *International Journal of Medical Informatics* **49** (1998) 217-229.

[13] V. Terziyan *at al.*, Intelligent Medical Diagnostics System Based on Integration of Statistical Methods. *Informatica Medica Slovenica* **3** (1996) 109-114.

[14] A. Tsymbal, S. Puuronen and V. Terziyan, Advanced Dynamic Selection of Diagnostic Methods. In: Proceedings of the CBMS'98, IEEE CS Press, Lubbock, Texas, 1998, pp. 50-54.

[15] D. Wolpert, Stacked Generalization. *Neural Networks* **5** (1992) 241-259.