Individualized Error Estimation for Classification and Regression Models

Krisztian Buza, Alexandros Nanopoulos, Lars Schmidt-Thieme

Abstract Estimating the error of classification and regression models is one of the most crucial tasks in machine learning. While the global error is capable to measure the quality of a model, local error estimates are even more interesting: on the one hand they contribute to better understanding of prediction models (where does and where does not work the model well), on the other hand they may provide powerful means to build successful ensembles that select for each region the most appropriate model(s). In this paper we introduce an extremely localized error estimation, called *individualized error estimation* (IEE), that estimates the error of a prediction model M for each instance x individually. To solve the problem of individualized error estimation, we apply a meta model M^*. We systematically investigate various combinations of elementary models M and meta models M^* on publicly available real-world data sets. Further, we illustrate the power of IEE in the context of time series classification: on 35 publicly available real-world time series data sets, we show that IEE is capable to enhance state-of-the art time series classification methods.

1 Introduction

Error estimation is one of the most crucial tasks in machine learning. For measuring the overall quality of a model, global error estimations are used, while for the task of analyzing the behavior of a model in different regions of the input space, local error estimations can be performed. This may be interesting on its own, as it contributes to the better understanding of prediction models. Furthermore, by allowing for the selection of the most appropriate model(s) for each region of the input space, local error estimation provides powerful means to build ensembles of classifiers or regressors.

Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany,
e-mail: {buza,nanopoulos,schmidt-thieme}@ismll.de

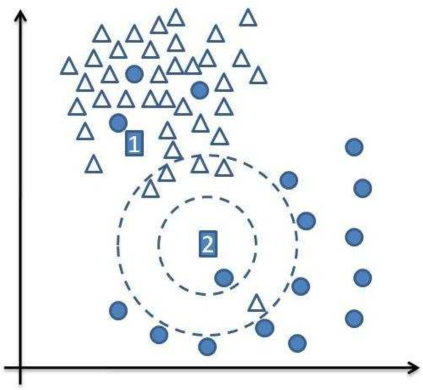

Fig. 1 A two-dimensional binary classification task. Assuming test instance '1' to be a triangle, and '2' to be a circle (ground-truth), we observe that 1-NN classifies '2' correctly, but misclassifies '1', while 6-NN outputs the appropriate class for '1', and misclassifies '2'.

In this paper we focus on local error estimation techniques. We introduce the notion of an extremely localized error estimation, called *individualized error estimation* (IEE), that estimates the error of a prediction model M for each instance x individually. To solve the problem of individualized error estimation, we apply a meta model M^*. We systematically investigate different combinations of elementary models M and meta models M^* on publicly available real-world data sets.

Furthermore, we show how to exploit IEE's power in the context of time series classification in order to enhance state-of-the art models. We evaluate our approach on 35 publicly available real-world time series data sets. The results show that our approach outperforms state-of-the art time series classification methods.

The paper is organized as follows. In section 2 we introduce individualized error estimation. In section 3 we systematically investigate IEE for various combinations of elementary models and meta models. In section 4 we describe an application of IEE to time series classification, we also present our experimental results. After summarizing related work in section 5, we conclude in section 6.

2 Individualized Error Estimation

We illustrate IEE in context of a simple binary classification task of a 2-dimensional data set. Figure 1 depicts a set of labeled instances from two classes that are denoted by triangles and circles. The density in the class of triangles (upper region) is larger than in the class of circles (lower region). We consider two test instances, denoted as '1' and '2', that have to be classified. We also assume that the ground-truth considers test instance '1' as a triangle, and '2' as a circle. Suppose, we use nearest neighbor (NN) models to classify the test instances.

Both for '1' and '2', the first nearest neighbor is a circle. Thus, the 1-NN classifies '1' incorrectly, while '2' is classified correctly. However, using e.g. a 6-NN classifier, due to the lower density in the circles' class, we observe '2' to be misclassified (see the large dashed circle around '2'), while '1' is classified correctly.

Table 1 Classification of test instances in the example

Instance	Ground-truth	Classification with 1-NN		Classification with 6-NN	
		M_1	M_1^*	M_2	M_2^*
1	triangle	circle	incorrect	triangle	correct
2	circle	circle	correct	triangle	incorrect

In this example we are concerned with two models: M_1: 1-NN and M_2: 6-NN. The perfect meta-model for M_1, denoted as M_1^*, would output that M_1 classifies the first test instance incorrectly, while it classifies the second test instance correctly (see 4th column of Tab. 1). Similarly, M_2^* (the perfect meta-model for M_2) would output that M_2 classifies the first test instance correctly, while it classifies the second test instance incorrectly (see the last column of Tab. 1).

In this simple example, the output of meta-models M_1^* and M_2^* consists of binary decisions whether the classifications by the elementary models M_1 and M_2 are correct or not. Please note, that one can develop more advanced meta-models, that output e.g. the likelihood (probability) of the correct decision, or, if we use a regression model at the elementary level, the meta-model could output the residuals (difference between predicted and true label).

Problem formulation Given a model M and a set of instances S (M predicts the labels of S), the task of Individualized Error Estimation (IEE) is to develop a meta-model M^* that is able to estimate the error of M for each instance of S individually.

Various versions of this task can be formulated (both in the context of classification and regression), and this defines the exact meaning of *error* in the above definition. Some of these possible versions include:

1. M is a classification model, and (as in the example above) the meta-model M^* takes binary decisions whether the classification by M is correct or not,
2. M is a classification model, and the meta-model M^* estimates for each instance the likelihood (probability) of the classification being correct,
3. M is a regression model, and M^* estimates the residuals (difference between predicted and true label) for each instance.

This generic definition allows for various classification and regression models, in this paper we are going to explore just a discrete set of models.

Our approach for IEE Fig. 2 summarizes the training procedure of our approach. Here, we describe the approach in the context of residual estimation, but it can simply be adapted for other versions of the task. The major steps are:

1. Split the labeled training data into two subsets D_A and D_B.
2. Train the elementary model M on D_A.
3. Let M predict the labels of D_B.
4. As the true labels of D_B are known, we can calculate the error of the predicted labels. In the case of residual estimation, when the labels are continuous, the

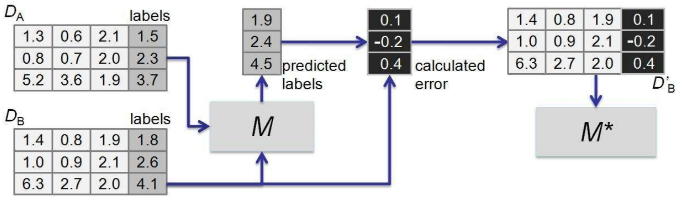

Fig. 2 Training procedure of our approach.

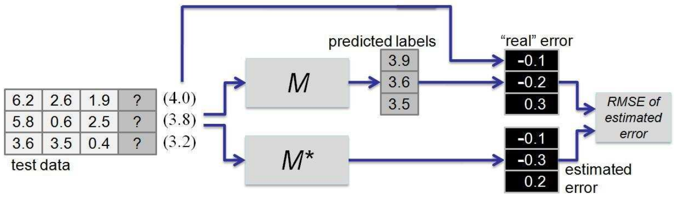

Fig. 3 Evaluation of IEE

calculated error $e(i)$ of an instance $i \in D_B$ is: $e(i) = M(i) - l(i)$, where $M(i)$ denotes the prediction of the elementary model M and $l(i)$ is the true label of i.
5. Train M^* on D_B using the calculated errors as labels.

When an unlabeled instance i' is processed, the elementary model M predicts its label, while the meta-model M^* estimates the error. In the case of residual estimation, the final predicted label of i' is: $l(i')^{final} = M(i') + M^*(i')$, where $M(i')$ denotes the prediction of the elementary model M, and $M^*(i')$ denotes the residual estimated by M^* for instance i'.

3 IEE with various models

Our approach (section 2), is generic as it allows to apply various classification and regression models both at the elementary level (as M) and at the meta-level (as M^*). Whenever a particular choice of M and M^* is made, we would like to evaluate how successfully M^* could predict the errors of M. As this evaluation procedure is non-trivial, we continue by describing our evaluation protocol.

In order to evaluate our approach (see Fig. 3), we first train both M, and M^* on training data as described in Section 2. Then, using M^*, we estimate the error for each instance of the disjoint test data D_{Test}. We also predict the labels of D_{Test} using M. Comparing the predicted labels to the ground-truth of the test data, we can calculate the true errors of the predicted labels. Finally, we compare the estimated errors to the true errors. When doing so, we calculate RMSE (root mean squared error) between the vector of true errors and the vector of estimated errors.

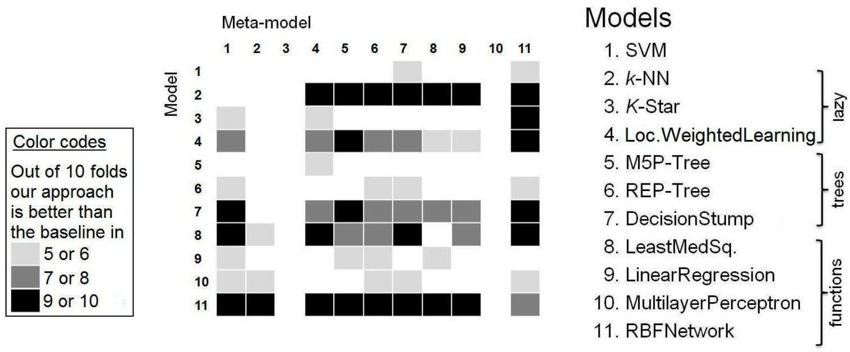

Fig. 4 Results: IEE with various combinations of models and meta-models (data: Communities). We used the same model types at the elementary and meta levels. A detailed description of the models can be found in the WEKA software package (Hall et al, 2009).

In our first experiment, we investigated various combinations of elementary models M and meta models M^* in the residual estimation setting as described above. We used the following publicly available real-world data sets from the UCI repository (Frank and Asuncion, 2010): i) Communities, ii) WineQuality (both red wines and white wines) and iii) Parkinson (both targets: motoric abilities and total abilities). As we observed very similar trends on all data sets, we only report the results on the Communities data set. As a simple baseline we used the meta-model M^*_{bl} that estimates that the prediction of the elementary model M was always correct.

We performed 10-fold-cross-validation: in each round the entire data is divided into 10 splits, out of which 1 serves as test data (D_{Test}), the other 9 splits are used as training data. Out of the 9 training splits, 5 splits constituted D_A and the remaining 4 splits belonged to D_B (see section 2 for D_A, D_B and the training procedure).

Fig. 4 summarizes our results: it shows for all the examined combinations of models and meta-models, in how many folds our approach was better than the baseline. The applied models are listed on the right of the figure, we used the implementations from the WEKA software package (Hall et al, 2009). In the matrix, the horizontal dimension corresponds to the applied meta-model M^*, while the elementary models are listed along the vertical dimension. For example, the 4th column position in the 2nd row corresponds to the combination where the model is k-NN and the meta-model is LWL. The color of the cell shows how many times (in how many rounds of the 10-fold-cross-validation) our trained meta-model M^* was better than the baseline M^*_{bl}. Black cells mean that M^* is better than M^*_{bl} (almost) always (10 or 9 times); dark gray cells indicate that M^* is better than M^*_{bl} 8 or 7 times, while light gray cells denote M^* being better than M^*_{bl} 6 or 5 times.

As a general observation, we see that for (almost) all of the elementary models, there are meta-models that are capable to deliver (relatively) good error estimations, although the particular choice of the meta-model is important: some of the meta-models deliver very poor estimations. As further observation, we see, that RBF-Networks and SVMs seem to work generally well as meta-models.

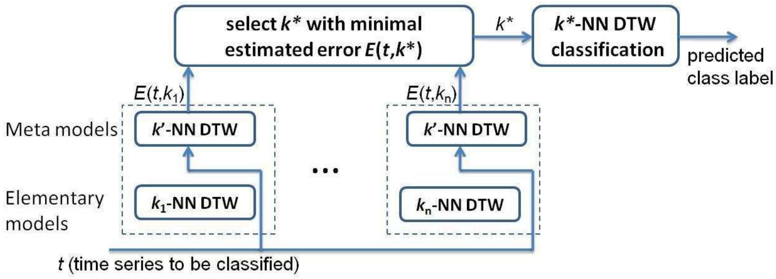

Fig. 5 IEE for time series classification

4 Time series classification using IEE

In our second experiment, we aimed at improving the accuracy of time series classification with IEE. In the time series domain, according to recent works, e.g. (Xi et al, 2006; Ding et al, 2008), simple nearest neighbor classifiers (using Dynamic Time Warping as distance measure) are generally very difficult to outperform (if it is possible at all). On the other hand, the ideal number of nearest neighbors, k, is non-trivial to be chosen and it may vary from region to region, just like in the example in section 2. Therefore, as elementary models M_i, we implemented DTW-based nearest neighbor classifiers for time series with odd k values between 1 and 10, i.e. M_1: 1-NN-DTW, M_2: 3-NN-DTW, ..., M_5: 9-NN-DTW. For each of them, we implemented a meta-model M_i^*, that aimed at estimating the likelihood of the error of M_i, $1 \leq i \leq 5$. This schema is depicted in Fig. 5. We trained the elementary classifiers M_i and the corresponding meta-models M_i^* as described in section 2.

Note, that in contrast to the previous experiment, where residuals were estimated, here, we apply IEE to estimate the error likelihood (see 2nd version of the IEE-problem in section 2). Therefore the training procedure is slightly different: when calculating the individual error of each predicted label in step 4 of our approach, as described in section 2, the error is considered as 0, if the classification produced by M was correct, otherwise the error is considered as 1. These calculated errors serve as meta-level labels. At the meta-level, we use nearest neighbor regression models, that simply average the meta level labels of the nearest neighbors. While the number of nearest neighbors at the elementary level is non-trivial to be selected, we observed that the meta-model for error likelihood estimation is much less sensitive to the number of nearest neighbors, k', and $k' = 5$ (almost) always leads to appropriate error likelihood estimation, therefore we fixed $k' = 5$ for all the meta-models M_i^*.

When classifying a test time series t, the meta-models estimated the error likelihood for each elementary model M_1: 1-NN-DTW, ..., M_5: 9-NN-DTW. Then we selected the elementary model with minimal estimated error likelihood and used it to classify t. (In this final step of predicting the class label of a test time series t, all the training data, $D_A \cup D_B$, is used by the selected nearest neighbor model.)

In our experiments we used 35 publicly available time series data sets from the collection used in (Ding et al, 2008). We omitted 3 data sets due to their tiny sizes:

Table 2 Summary of our results for time series classification: number of cases where our approach wins / looses (significantly wins / looses in parenthesis) against the baselines.

	$p = 1\%$	$p = 5\%$	$p = 10\%$	total
Wins against 1-NN-DTW	30 (20)	34 (29)	34 (31)	98 (80)
Looses against 1-NN-DTW	5 (1)	1 (0)	1 (0)	7 (1)
Wins against *k*-NN-DTW	30 (15)	30 (9)	28 (14)	88 (38)
Looses against *k*-NN-DTW	5 (1)	5 (1)	7 (1)	17 (3)

each of them contained less than 100 observations. We performed 10-fold-cross-validation. As baselines we used 1-NN-DTW and k-NN-DTW with globally best k selected on a hold-out subset of the training data (after selecting the globally optimal k, all the training data is used to classify test time series). We tested statistical significance using t-test at level 0.05. For the original data sets, in the majority of the cases, we did not observe significant differences (often all approaches, ours and both baselines performed very well). As recent research showed, bad hubs[1] are responsible for a surprisingly high portion of the error (Radovanovic et al, 2010). Thus, in order to make the task more challenging, we artificially introduced some bad hubs: we changed the best hubs to bad ones (set the class labels to an artificial 'noise' class) by infecting in total p % of the entire data set. Tab. 2 summarizes our results for 3 different levels of p. In the table we report in how many cases our approach wins/looses against the baselines, in parenthesis we report in how many cases the wins/looses are statistically significant.

5 Related work

Error-prediction methods are usually applied globally in order to estimate the overall performance of a classification model (Molinaro et al, 2005; Jain et al, 1987). Closely related to ours is the work of Tsuda et al. (Tsuda et al, 2001), who proposed an individualized approach for predicting the leave-one-out error of vector classification with support vector machines (SVM) and linear programming machines (LPM). Compared to this work, our proposed approach is more generic, as i) we did not only focus on the overall leave one out error, ii) we did not only focus on vector classification (but also allow for more complex structures like time series), and, most importantly, iii) in the current work we have shown how to exploit IEE to enhance classification.

IEE is also related to boosting, where residuals are estimated in order to enhance classification like in (Duffy and Helmbold, 2002). However, IEE is not limited to the

[1] Hubs are time series that appear most frequently as nearest neighbors of other time series. Denote the set of time series for which t is the nearest neighbor as N_t. A hub t is a bad hub if its class label is different from the class labels of *many* time series in N_t. See also (Radovanovic et al, 2010).

estimation of residuals and in this sort of sense IEE is more generic than boosting. Moreover, in the case of boosting a (long) series of (usually weak) models of the same type is used, whereas in IEE we use a pair of models: an elementary model and a meta model, and the models may belong to different (strong) model types.

The presented application of IEE (estimation of the *k* for time series classification using *k*-NN-DTW) is related to locally adaptive models (Hastie and Tibshirani, 1996; Domeniconi and Gunopulos, 2001; Domeniconi et al, 2002). In contrast to these works, our approach adapts by selecting the proper value of *k* and not by determining a localized distance function.

6 Conclusion

In this paper we introduced the notion of individualized error estimation (IEE) and defined three versions of the IEE-problem. We systematically investigated IEE in context of various prediction models. We observed RBF-Networks and SVMs to deliver good error estimations compared to the other examined models. We have also shown that IEE is capable to enhance state-of-the art time series classification models. As future work, we aim at exploring new IEE-problem and applications.

References

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: Experimental comparison of representations and distance measures. VLDB Endowment 1(2):1542–1552

Domeniconi C, Gunopulos D (2001) Adaptive nearest neighbor classification using support vector machines. Advances in NIPS 14:665–672

Domeniconi C, Peng J, Gunopulos D (2002) Locally adaptive metric nearest-neighbor classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(9):1281–1285

Duffy N, Helmbold D (2002) Boosting methods for regression. Machine Learning 47:153–200

Frank A, Asuncion A (2010) UCI machine learning repository. Tech. rep., University of California, School of Information and Computer Sciences, Irvine, URL http://archive.ics.uci.edu/ml

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: An update. SIGKDD Explorations 11(1):10–18

Hastie T, Tibshirani R (1996) Discriminant adaptive nearest neighbor classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(6):607–616

Jain AK, Dubes RC, Chen CC (1987) Bootstrap techniques for error estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 5(9):606–633

Molinaro AM, Simon R, Pfeiffer RM (2005) Prediction error estimation: a comparison of resampling methods. Bioinformatics 21(15):3301–3307

Radovanovic M, Nanopoulos A, Ivanovic M (2010) Time-series classification in many intrinsic dimensions. In: Proc. 10th SIAM International Conference on Data Mining, SIAM, pp 677–688

Tsuda K, Rätsch G, Mika S, Müller KR (2001) Learning to predict the leave-one-out error of kernel based classifiers. ICANN 2001, LNCS 2130/2001:331–338

Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: Proc. 23th Int'l. Conf. on Machine Learning, ACM, pp 1033–1040