Regression as classification

Conference Paper · March 2012

DOI: 10.1109/SECon.2012.6196887

2 authors:

Raied Salman
American College of Commerce and Technology
16 PUBLICATIONS **184** CITATIONS
SEE PROFILE

Vojislav Kecman
Virginia Commonwealth University
118 PUBLICATIONS **3,996** CITATIONS
SEE PROFILE

Regression as Classification

Raied Salman
Computer Science Department, Virginia Commonwealth University
Richmond, VA 23284-3068, USA
salmanr@vcu.edu

Vojislav Kecman
Computer Science Department, Virginia Commonwealth University
Richmond, VA 23284-3068, USA
vkecman@vcu.edu

***Abstract*—The paper presents how solving regression problems can be posed as finding solutions to multiclass classification tasks. The accuracy (averaged over several benchmarking data sets used in this study) of an approximating (hyper)surface to the data points over a given high-dimensional input space created by a nonlinear multiclass classifier is slightly superior to the solution obtained by regression (hyper)surface. In terms of the CPU time needed for training i.e., for tuning the hyperparameters of the models, the nonlinear classifier shows significant (order of magnitudes for large datasets) advantages too. Here, the support vector machine (SVM) has been solving given regression problems as a classic SVM regressor and as the SVM classifier. In order to transform a regression problem into a classification task two possible discretizations of a continuous output (target) vector y are presented and compared. A very strict double (nested) cross-validation technique has been used for measuring performances of regression and multiclass classification SVMs. A novel approach and the experimental results obtained for five benchmarking regression data sets warrant both further theoretical investigations and broad application in practice.**

Keywords-Classification; Regression; Support Vector Machine; Machine-Learning; Data mining; Discretization

I. INTRODUCTION

Machine learning (ML) techniques are used for solving various problems in contemporary science, medicine, finance, engineering and many other areas. There is no field of human activities untouched by ML tools (a.k.a. neural networks, support vector machines, data mining, and/or knowledge discovery, etc). In particular, ML tools are aiming at solving two classic statistical tasks – classification (pattern recognition) and regression (function approximation). Here we will focus on presenting a novel approach in solving regression problems as multiclass classification tasks. Recently, in [1], it has been shown how one can solve regression problems posing them as the multiclass classification tasks. Very good results of a multiclass approach averaged over four benchmarking regression data sets have been presented. The multiclass classifier used in that paper has been the Adaptive Local Hyperplane (ALH) introduced in [2]. The choice of the ALH was natural because, in [2], it showed extremely good results on eleven standard benchmarking classification data sets, outperforming seven other machine learning tools including KNN classifier, SVMs and *K*-local hyperplane (linear manifold) distance nearest neighbor (HKNN) algorithm. The results presented in [1] were a big encouragement and provided a strong motivation for exploring the approach of transforming regression problems into the classification ones within a much stricter experimental approach.

The topic here is to investigate the very characteristics of the proposed approach and to do a strict comparison of a nonlinear (NL) regressor and an NL multiclass classifier in solving regression problems. In order to do fair comparisons, SVMs are used for solving both tasks - regression and multiclass classification. The easily available and the most popular benchmarking SVM tool LibSVM [3] was used in all experiments.

As for the model comparisons environment, the double (nested) cross-validation (resampling) has been implemented here. Note that all results presented in [1] have been obtained by using a standard single (10-fold) cross-validation. This was a fair approach for validating whether the idea of transforming regression problems into the classification tasks is feasible at all. However, such an approach is not quite valid for comparing different models because in a single cross-validation all the data points available are used for both hyper-parameter determination and accuracy estimation. While such a procedure is good for estimating the hyperparameters of a single model it does not produce a true estimate of the accuracy for the future, unseen during the training, data pairs. If the main goal is to compare different ML models on the same data sets and under same conditions, double cross-validation must be used [4].

The double cross-validation is a very rigorous scheme for assessing a model's performance [4]. Here, we evaluate the generalization performances of SVM regressor and SVM multiclass classifiers by using the double cross-validation procedure. The double resampling procedure is structured as the two loop algorithm. In the outer loop, the data set is separated into J_1 roughly equal-sized parts (in this paper J_1 = 10 was used in all the experiments). Each part is held out in turn as the test set, and the remaining 9 parts are used as the training set. In the inner loop, J_2-fold cross-validation is performed over the training set only to determine the best values of hyper-parameters (here, J_2 = 10). The best model obtained in the inner-loop is then applied on the test set. The double cross-validation procedure ensures that the class labels of the test data won't be seen when tuning the hyper-parameters, which is consistent with the real-world scenario. Obviously such a rigorous procedure is done in many runs. In our case here, for the Gaussian kernel SVMs and when solving a regression problem which has three hyperparameters (penalty parameter C, shape parameter σ, and the size of ε-tube), the number of iteration runs increases very quickly as follows: suppose we want to select the best parameter out of 5

Corresponding author's Email: *vkecman@vcu.edu*

predefined ones for each hyperparameter of an SVM regressor. This amounts to 10 x 10 x 5 x 5 x 5 = 12,500 runs over the data set. All in the field of machine learning may well be aware that giving only 5 values for the hyperparameters is usually a scarce approach. In praxis one typically uses much bigger sets of values for each hyperparameter in order to find the best one, and this then leads to the significant training time. In this paper, the training CPU times for SVM regressors and SVM multiclass classifiers are also compared, showing multifold speeding up when solving regression as classification for large datasets (see the times in Tables 1 and 2 needed to train the prototask data here).

In terms of the accuracy, the results in solving five benchmarking regression tasks shown here will present SVM regression and classification algorithms as strongly competing models where each approach shows merits for a specific class of high-dimensional function approximation problems.

The rest of the paper is structured as follows. Section 2 repeats the basics of the SVM algorithms for regression and classification problems and it points to the major differences in the algorithm. It also presents the two ways of a discretization as the tools for transforming regression problems into the multi-class classification tasks. Section 3 introduces five well known regression problems (Boston housing data, concrete compressive strength, servo data, prototask (comp-active) data and wine data). It also presents performances of SVMs in solving the given regression problems in two ways; as the multi-class classifier and as a regressor. The first three data sets and the wine data set are downloadable from the UC Irvine Machine Learning Repository [5] while the prototask data set can be downloaded from The University of Toronto Delve site as given in [7]. In addition, section 3 discusses some theoretical properties of the two approaches in solving regression and classification problems related to the bounds. After the concluding section 4, the paper finishes with relevant references.

II. TRANSFORMING REGRESSION INTO CLASSIFICATION

Regression problems are defined as follows: l instances (samples, measurements) with d input features are given as the input points or vectors $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T$, each $\mathbf{x}_i$ is associated with known output value y_i, and $y_i \in \Re^1$. The difference in respect to the classification problems is that the output values are real numbers in regression problems. However, this setting can also be looked at as the classification problem where each y_i is treated as the 'class label'. Hence, each regression problem is a multiclass classification problem with maximally l classes. Relabeling continuous y_i values into the classes is readily done after sorting y_i and then just orderly assigning the labels. However, this obviously leads to several undesired consequences, the most important one being that we don't want to model the noise always present in data and we would like to control the variance of the model. In order to filter the noise out, as well as to reduce the variance of the model, the output vector $\mathbf{y}$ is approximated by the SVM regressor after defining the so-called 'ε-insensitivity zone' (a.k.a. ε-tube). The 'weights' of the SVM model are obtained by usually finding the dual variables (α_i in classification, and both α_i and α_i^* in regression, $i = 1, l$) first. More precisely the SVM models are defined as

$$f(\mathbf{x}) = \sum_{i=1}^{N_{SV}} w_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{1}$$

where, in classification,

$$w_i = \alpha_i y_i \tag{2}$$

and in regression,

$$w_i = \alpha_i - \alpha_i^* . \tag{3}$$

While the two final models look alike, the corresponding dual Lagrangian (QP) problems to be solved for α-s are fairly different:

in classification,

$$L_{\mathrm{d}}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) ,$$

s.t.

$$0 \le \alpha_i \le C, \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0 \;\; i = 1, \ldots, l , \tag{4}$$

and in regression,

$$L_{\mathrm{d}} = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) y_i - \varepsilon \sum_{i=1}^{l} \left(\alpha_i + \alpha_i^*\right)$$

$$-0.5 \sum_{i=1}^{l} \sum_{j=1}^{l} \left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right) K(\mathbf{x}_i, \mathbf{x}_j) ,$$

s.t. $\quad 0 \le \alpha_i \le C, 0 \le \alpha_i^* \le C,$

and

$$\sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) = 0 \;\; i = 1, \ldots, l \tag{5}$$

The fundamental difference in solving the QP problems (4) and (5) is in the size of the corresponding Hessian matrix $\mathbf{H}$. In classification $\mathbf{H}$ is an (l, l) matrix, while in regression it is an ($2l$, $2l$) one. Hence, it's of a double size and inherently nonsingular.

Here, we will attempt something else. The regression problems will be transformed into the multiclass classification tasks by a discretization.

There are several ways in which the discretization can be performed. The obvious and direct way of transforming the regression problem into the classification task is to perform the discretization of the target vector $\mathbf{y}$ by a fixed ε-tube size into a set of N classes (as already mentioned above). The size of the ε-insensitivity zone controls the accuracy of the approximation and this step is similar to defining the ε-tube in the SVMs for a regression. However, unlike in SVMs regression problem defined by (5), the SVM classifier (obtained after the discretization) solves the multiclass classification task posed as in (4). Fig. 1 shows the simple example; a one-dimensional noisy hyperbolic function (left) and the class label of each training sample after using ε = 0.0931 i.e., after the

discretization of **y** into $N = 10$ classes (right). The solid curve shown in both graphs is a noiseless function. Dashed lines in the right hand graph show class boundaries and the mean value of each class is shown as a solid line (the mean value of class 2 is explicitly pointed at with a text over it).

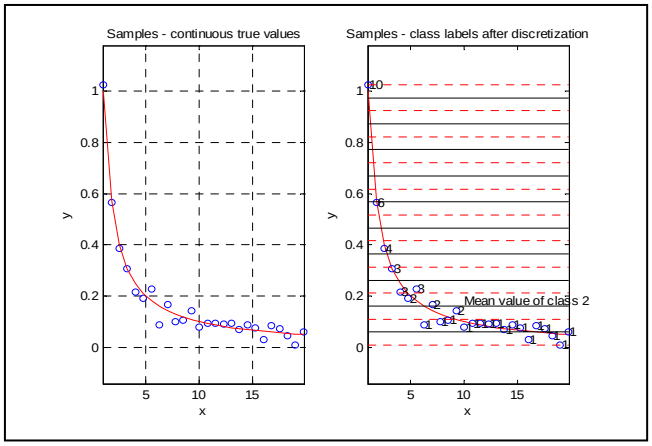

Figure 1. Class assignments in a regression task with equal ε-tube

The procedure for transforming the continuous values of the response (output) vector **y** into the set of labels is the same for any high-dimensional $\Re^d \rightarrow \Re^1$ mapping. The $\Re^1 \rightarrow \Re^1$ regression example shown in Fig. 1 is used for simplicity of visualization only. After the discretization, a vector of N classes' mean values is saved. In the example above this vector $\boldsymbol{\mu} = [0.0592 \quad 0.1608 \quad 0.2624 \quad 0.3640 \quad 0.4656 \quad 0.5672 \quad 0.6689 \quad 0.7705 \quad 0.8721 \quad 0.9737]^T$.

The SVM multiclass classification produces the vector $\mathbf{y}_p$ containing the predicted class labels which are then replaced with the mean values of each class. This works as follows in Fig. 1; suppose that the ALH classifier would predict the belongings of the 8 data points to be $\mathbf{y}_p = [4 \ 3 \ 3 \ 3 \ 2 \ 1 \ 1 \ 1]^T$. These labels would then be translated into the following 8 final approximated values of the noisy data points $\mathbf{y}_a = [0.3640 \ 0.2624 \ 0.2624 \ 0.2624 \ 0.1608 \ 0.0592 \ 0.0592 \ 0.0592]^T$.

Note that a discretization into N classes with an ε-tube equal over the whole range leads to the empty classes. Here, classes 5, 7, 8 and 9 don't have any data points. In other words, later, a reassigning of predicted class labels as the continuous values will lead to higher accuracies in the flat portions of the function. Finer discretization increases the accuracy but it may also result in an over-fitting behavior of the model. The right level of a discretization (i.e., the best number of classes N, which corresponds to the best size of the ε-tube in an inverse way) should be determined by *cross-validation*. The same is valid for the penalty parameter C, and the shape parameter γ (if Gaussian kernel is used which is the case here). Hence, there are three hyper-parameters in the SVM multiclass classification which should be determined by cross-validation. Remind that in SVM regression the same three hyper-parameters must be tuned, but the size of the Hessian **H** is doubled and much more badly conditioned than in the classification case.

A slightly better way of discretization is achieved by specifying the *minimal size of the* ε-tube. For a given minimal ε-tube, during the discretization, the care is taken that in each class there is at least one training data point. The right minimal size of the ε -tube should also be determined by the cross-validation. Obviously, this leads to the varying sizes of the tubes for different classes but it improves the accuracy by avoiding grouping too many data in the flat part of the regression (hyper)surface. The results of the second type of discretization are shown in Fig. 2.

The third possible way to do the discretization of continuous values is to divide them by the so-called equal-frequency discretization. This is done by dividing the output vector **y** value range into a number of intervals (classes) so that (approximately) the same number of training data points are in each interval (class). For example, if one chooses to perform the discretization into $N = 10$ classes, each interval will contain about 10% of the training data. However, such a discretization didn't lead to good results and it is no longer considered here.

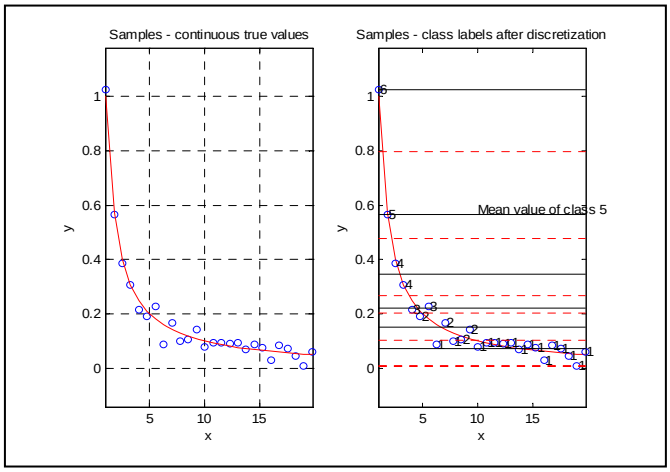

Figure 2. Class assignments in a regression task with varying ε-tube

Once the approximation $\mathbf{y}_a$ to a test vector **y** is calculated, the relative percentage error of the approximation is calculated by using the following expression,

$$e = 100 \frac{\|\mathbf{y} - \mathbf{y}_a\|_2}{\|\mathbf{y}\|_2} \qquad (6)$$

III. Regression Data Sets and The Results

In order to show the performance of the proposed approach in solving regression tasks, five benchmarking regression data sets are selected for the study here. They are *Boston housing* data [5, 6] ; *prototask* (*comp-active*) data base [7] ; *concrete* compressive strength [5, 8] ; *servo* data and *wine* data [5].

The details of all five data sets can be found in the references given and the simulation results are shown in Table 1. The basic characteristics of the data sets used are as follows: *Boston housing* is a collection of 506 samples with a 12 dimensional input vectors, *prototask* data set contains 8192 instances with 21-dimensional input vectors, *concrete* compressive strength data consists of 1030 samples with 8 dimensional inputs, the *servo* data set contains only 167

samples with 4-dimensional feature vectors and the *wine* data is a collection of 1599 12-dimensional feature vectors. Note that the wine data's outputs are integer values corresponding to 6 types of wine but in the references, the wine data set is treated as the regression problem. All five data sets are used as the regression benchmarking data sets in many investigations and papers, most of which can be found at the UCI site [5].

TABLE I. PERCENTAGE ERRORS AND CPU TIMES FOR 5 DATA SETS OBTAINED BY TWO SVM CLASSIFIERS AND SVM REGRESSOR

Data/Algorithm	**SVM classifier 1** *varying ε-tube*		**SVM classifier 2** *fixed ε-tube*	
	Error %	***CPU time*** ***sec/run***	***Error*** %	***CPU time*** ***sec/run***
Boston Housing	**12.6**	0.929	12.8	0.626
Prototask	3.22	428	**3.2**	**98**
Concrete	14.7	7.49	14.7	**6.79**
Servo	**19.6**	**0.054**	20	0.0815
Wine	**11.8**	3.73	11.9	4.88
Average Error	**12.4**		12.5	

Data/Algorithm	**SVM Regressor**	
	Error %	***CPU time sec/run***
Boston Housing	12.8	**0.475**
Prototask	3.3	345,600.00
Concrete	**13**	16.3
Servo	24.9	0.426
Wine	**12**	**2.64**
Average Error	13.2	

a. The two classifiers differ only in the way how the discretization of a continuous **y** has been done

Here we have compared the performances of three SVM models, all of which use Gaussian kernels - two classification SVM models are solving regression problems as the multiclass classification tasks and one SVM regression model is a standard SVM regressor. The two SVM classifiers differ only in the way the discretization (described above) has been carried out.

There are several things to point at in Table 1. First and foremost, in solving regression problems, the SVM classifiers have slightly outperformed the SVM regressor. Furthermore, the SVM classifier 1 (with the varying ε-tube) has shown the best overall performance averaged over all five data sets. It is also a winner in terms of the accuracy for 3 out of 5 data sets. As for the CPU time needed to finish the training, SVM classifiers have shown better performances as well. This is particularly obvious for the largest data set *prototask* when SVM regressor needed four days to finish the training. Generally, once the data set goes over several thousands of training samples the training speed-up of the SVM classifiers becomes obvious. (Faced with a regression problem having more than 10,000 samples excludes solving at as the regression tasks for the training CPU time needed). The second best in terms of both accuracy and time is the SVM classifier 2)which works with the fixed ε-tube). Note that all the results have been obtained within the double (nested) cross-validation experimental procedures, guaranteeing the most objective results and the true performance range on the future data samples. Also note that the LibSVM software has been used and the same number of hyperparameters has been tested in all the simulations; ($C = \{10^{-3}\ 10^{-2} \ \ldots\ 10^{4}\}$, $\sigma = \{10^{-2}\ 10^{-1} \ \ldots\ 10^{2}\}$, $N = \{2{:}2{:}100\}$, $\varepsilon = \{0.005$ to $0.2\}$ of $\max(\mathbf{y})$).

There is one more intriguing and obvious issue raised by the results in Table 1. Namely, both the SVM classifiers and the SVM regressor are competing tightly for the primacy in solving the regression problems used here by showing distinctly different performances in terms of the accuracy for the two types of the function approximation problems. SVM classifiers show better results for the *prototask* and *servo* data and slightly better accuracy than the SVM regressor for the *Boston housing data* (actually accuracies are almost equal). Equal results are also achieved for the *wine* data set, while the SVM regressor displays better performance for the *concrete* data set. Some explanations and hints about these results are related to the very nature of the regression problems and they may be obtained by looking at the target values y_i and their distributions as shown in Figs 3 and 4 respectively.

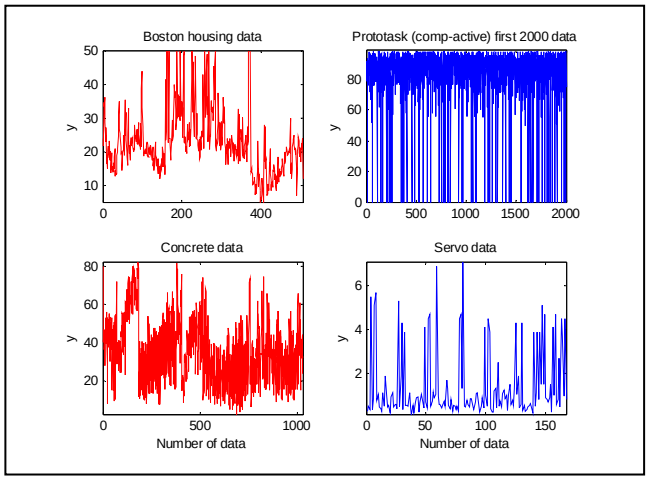

Figure 3. Output values in four regression problems

It seems that the SVM regressor suits better regression problems where the target values are 'uniformly' distributed over the range (left columns of graphs in Figs. 3 and 4), while in the problems *prototask* and *servo,* where the SVM classifiers show clear advantages, the output vector **y** shows very 'spiky' behavior covering primarily either upper part of the range (*prototask*) or the lower one (*servo*) and making irregular spikes to the lower values (*prototask*) i.e., higher ones (*servo*). (Note that only the first 2,000 *prototask* data are shown for the sake of a better insight into the nature of a target vector **y**. The very same character of the *y* signal stretches to the end of a

target vector **y**). Sorted output values y_i for the same regression data are shown in Fig. 4.

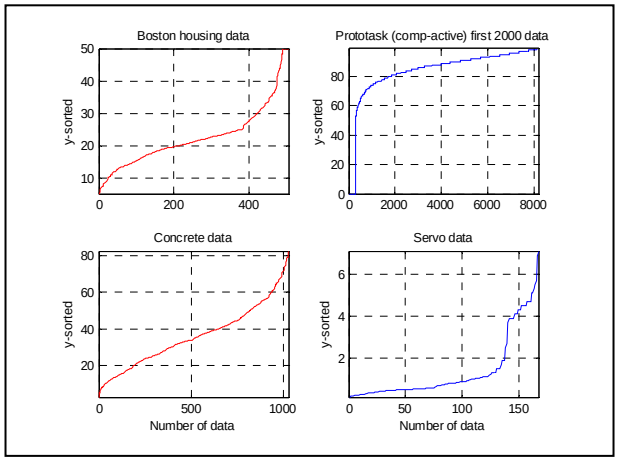

Figure 4. Sorted output values in four regression problems

The 'spiky' functions (*prototask* and *servo*) resemble the Poissonian, or exponential, distribution, in which a great deal of target values y_i will after the discretization fall into a small number of not-well-balanced classes. The experimental results show that when faced with the 'spiky' nature of the regression problems, a discretization and multi-class classification through the use of SVMs will most likely lead to the better function approximation.

Here, it should also be mentioned that we have also run the experiments where the discretization of the output vector **y** values has been done by using *k*-means clustering. Results obtained are close to the SVM classifiers 1 and 2 and for the sake of the constrained space they are not shown here.

There is one more challenging problem (actually a set of questions and some hints regarding the theoretical properties of the novel approach in solving regression problems) raised by the results obtained, which is beyond the scope of this paper. Basically, it concerns the theory of bounds for both the multi-class classification and regression problems. (The next remarks follow strictly [9]). First, there is a basic question which arises naturally from the results obtained and it is whether, in general, the bounds on classification are tighter than the ones in the regression. (If so, in addition to showing better experimental results, one may get strong theoretical support for using a classification approach when solving regression tasks). One may also be interested in some other theory-based questions, such as whether it is easier to prove classification bounds or the regression ones.

Some possible hints to the answers (repeating some arguments from [9]) are as follows: The very bounds we are talking about here usually refer to comparing the empirical error of our decision function with its true error. (The other types of bounds used are usually called oracle inequalities). Since typically, the function classes in regression are somewhat larger by the nature of the loss function, bounds in regression are often worse. But there are also exceptions, in particular, if the best predictor is in the hypothesis class and the loss is sufficiently convex (e.g. square loss). In this case, regression bounds may be better. Note however that for general regression data sets it may be highly unlikely to get the predictor in the hypothesis class.

As for the easiness of proving the bounds, one thing is certain - good bounds are always hard to prove, independent of whether they are for the classification or regression. Thus, one may state the arguments quite differently, and compare the combinatorial bounds (in classification) vs. continuous ones (in regression). If the sharpness is not of primary interest, then generic off-the-shelf bounds are typically straight-forward to apply, and the only tricky part is the used complexity measure (VC-dimension, covering numbers, Rademacher averages, ...). A somewhat unifying approach that can be implemented is the McDiarmid inequality, which is based (after symmetrization) on Rademacher averages. The latter can be bounded by several concepts such as VC dimension or covering numbers. In the future, while answering the questions raised here, some techniques along the lines of those above can be found in the book on density estimation [10]. As for the bounds in the multi-class classification problems (which is the problem to solve after discretization of the continuous output values y_i), multiclass classification also requires a loss function, and in most cases the above techniques should work. This is most likely the reason why nobody looked deeper into that question until now. The issues of the bounds will be a matter of further theoretical investigations of the approach for solving regression problems as the classification tasks proposed in this paper and their solutions will shed more lights to the questions whether to solve regression problems as a regression or multiclass classification tasks.

IV. Conclusions

This paper shows how the regression problems, after discretization, can be solved as multiclass classification ones. SVM classifier and regressor are the modeling tools here. Two slightly different ways of discretizations are introduced and compared. On average, SVM classifier with varying ε-tube shows the best results. It also shows, in the case of 3 out of 5 regression problems, the best accuracy. In addition to being better regressors (although the differences in accuracy are not too big), the SVM classifiers are also superior in terms of CPU times needed for training SVM. This is particularly pronounced when the number of training data surpasses several thousands samples. In such cases one should rely on solving regression problems as the multiclass classification tasks. Based on the general properties that the bounds in regression are often worse than the ones in classification, the proposed approach of solving regression problems as classification ones may seem very attractive from the theoretical point of view as well. This may then well be one of the most important explorations in the future. The model comparisons have been done by implementing the double (nested) cross-validation i.e., resampling, structured as the two loop algorithm, which is a very rigorous scheme for assessing models' performances. In such an experimental environment, not all the data points available are used for both hyper-parameter determination and accuracy estimation simultaneously. On the contrary, data for

the hyperparameter selection are strictly separated from the data for the accuracy estimation, making in this way double cross-validation the only objective environment for the fair comparisons of different models. Thus, if the main goal is to compare different ML models on the same data sets and under the same conditions, double cross-validation must be used, as was done here.

Acknowledgments

The authors would like to thank Prof. I. Steinwart for his hints and help in discussing the bounds for classification and regression.

REFERENCES

[1] Kecman, V., Yang, T., Adaptive Local Hyperplane for Regression Tasks, Proc. of IJCNN 2009, IEEE International Joint Conference on Neural Networks, pp. 1566 – 1570, June 14 – 19, Atlanta, GA, USA, 2009

[2] Yang, T., Kecman, V., "Adaptive Local Hyperplane Classification", *Neurocomputing*, 71, pp.3001-3004, 2008.

[3] Chang, Hsu C-W, C-C, Lin C-J, A Practical Guide to Support Vector Classification. Technical Report Department of Computer Science and Information Engineering, National Taiwan University, 2003

[4] Yang, T. and Kecman, V., Machine Learning by Adaptive Local Hyperplane Algorithm: Theory and Applications, VDM-Verlag, Saarbrücken, Germany, 2010

[5] Asuncion, A., Newman, D. J., "UCI ML Repository", [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA, University of California, School of Information and Computer Science, 2007.

[6] Harrison, D., Rubinfeld, D. L., "Hedonic prices and the demand for clean air", *J. Environ. Economics & Management*, vol.5, pp. 81-102, 1978.

[7] Delve, Data for Evaluating Learning in Valid Experiments, [http://www.cs.toronto.edu/~delve], The University of Toronto, Toronto, Canada, 2008

[8] Yeh, I-C., "Modeling of strength of high performance concrete using artificial neural networks", *Cement and Concrete Research*, Vol. 28, No. 12, pp. 1797-1808, 1998.

[9] Steinwart, I., Personal communication, 2011

[10] Devroye, L., Lugosi, G., Combinatorial Methods in Density Estimation, Springer-Verlag, 208 p., 2001