Instance Selection in Logical Rule Extraction for Regression Problems

Mirosław Kordos[1], Marcin Blachnik[2], Szymon Białka[1]

[1] University of Bielsko-Biala, Department of Computer Science and Automatics, Bielsko-Biała, Willowa 2, Poland: mkordos@ath.bielsko.pl
[2] Silesian University of Technology, Department of Management and Informatics, Katowice, Krasinskiego 8, Poland; marcin.blachnik@polsl.pl

Lecture Notes in Artificial Intelligence, Vol. 7895, pp. 167-175, ICAISC, June 2013 .
The original publication is available at www.springerlink.com

Abstract. The paper presents three algorithms of instance selection for regression problems, which extend the capabilities of the CNN, ENN and CA algorithms used for classification tasks. Various combinations of the algorithms are experimentally evaluated as data preprocessing for regression tree induction. The influence of the instance selection algorithms and their parameters on the accuracy and rules produced by regression trees is evaluated and compared to the results obtained with tree pruning.

1 Introduction

1.1 Instance Selection in Classification Problems

The reasons for reducing the number of instance in the training set include: noise reduction by elimination outliers, reducing the data set size and improving generalization by eliminating instances that are too similar to each other, faster training of the model an a smaller dataset and faster prediction in of the model, especially in the case of lazy-learning algorithms, as k-NN. The early research in that area of instance selection in classification tasks lead to the Condensed Nearest Neighbor rule (CNN) [1] and Edited Nearest Neighbor rule (ENN) [2]. Another algorithm called CA, which can be thought of as an improved version of CNN was proposed by Chang [3]. In the following years, other, more complex algorithms were developed, such as Drop1-5 [4], IB3, Gabriel Editing (GE) and Relative Neighborhood Graph Editing (RNGE), Iterative Case Filtering (ICF), ENRBF2, ELH, ELGrow and Explore [5]. A large survey including almost 70 different algorithms of instance selection for classification tasks can be found in [6]. An interesting idea was proposed by Jankowski and Grochowski in [7]; to use the algorithms as instance filters for other machine learning algorithms like SVM, decision trees etc. By filtering noisy and compacting redundant examples they were able to improve the quality and speed of other classification algorithms.

1.2 Challenges in Regression Tasks

The instance selection issue for regression tasks is much more complex. In classification tasks only the boundaries between classes must be determined, while in regression tasks the output value must be assessed at each point of the input space. Moreover, in classification tasks there are at most several different classes, while in regression tasks, the output of the system is continuous, so there is an unlimited number of possible values to be predicted by the system. That is the reason why the dataset compression obtained by instance selection can be much higher in classification than in non-linear regression problems. The decision about rejection of a given vector in classification tasks can be made based on a right or wrong classification of the vector. In regression problems, rather a threshold defining the difference between the predicted and the actual value should be set. Determining the threshold (which is rather a function than a constant value) is an issue specific only to regression tasks. Another issue is the error measure, which in classification tasks is very straightforward, while in regression tasks, it can be defined in several ways and in practical solutions not always the simple error definitions as the MSE (mean square error) work best [8]. Because of the challenges, there were very few approaches in the literature to instance selection for regression problems. Moreover, the approaches were verified only on artificial datasets generated especially for the purpose of testing the algorithms. Zhang [9] presented a method to select the input vectors while calculating the output with k-NN. Tolvi [10] presented a genetic algorithm to perform feature and instance selection for linear regression models. In their works Guillen et al. [11] discussed the concept of mutual information used for selection of prototypes in regression problems.

1.3 Decision Trees

Clear logical rules are crucial factors in some practical implementations, where the decision must not only be made by the system but it also must be explained to humans [8, 12]. The advantage of regression trees is their ability to easily generate comprehensive logical rules from data in a way easily to understand. For that reason in our practical implementations of computational intelligence to technological process optimization we always used a decision tree, usually as one of the components of a hybrid model. To keep the logical rules simple we use a univariate tree in all non-terminal nodes. Only the value assigned to the terminal leaves can be either a mean value of the vectors in the leaf or a linear regression, which includes no more than three most significant features in the leaf. Instance selection influence on classification trees were studied in [13] and [14]. In the current work we study that in case of regression trees.

2 Instance Selection Algorithms

2.1 ENN, CNN and CA Instance Selection Algorithms for Classification Problems

The CNN (Condensed Nearest Neighbor) algorithm was proposed by Hart [1]. The purpose of CNN is to reject these instances, which do not bring any additional information into the classification process. The algorithm starts with only one randomly chosen

instance from the original dataset T. And this instance is added to the new dataset P. Then each remaining instance from T is classified with the k-NN algorithm, using the k nearest neighbors from the dataset P. Only if the classification is wrong - the instance is added to P. The ENN (Edited Nearest Neighbor) algorithm was created by Wilson [2]. ENN works as a noise filter. The main idea of the algorithm is to remove a given instance if its class is different than the majority class of its neighbors. ENN starts with the whole original training set T. Each instance, which is wrongly classified by its k nearest neighbors is removed from the dataset, as it is supposed to be an outlier. In repeated ENN, the process of ENN is iteratively repeated as long as there are any instances wrongly classified. In all k-NN algorithm, the ENN is repeated for all k from $k = 1$ to k_max. The CA algorithm [3] works in a similar way to CNN. However, instead of rejecting one example, it replaces two closest examples of the same class by a single example situated in the middle of them. If this does not decrease the classification accuracy, the change is kept, otherwise it is rejected. Then the next pair of closest examples is considered. The algorithm works iteratively as long as there are no more examples to merge without affecting the classification accuracy.

2.2 RegENN, RegCNN and RegCA Instance Selection Algorithms for Regression Problems

Algorithm 1 RegENN algorithm

Require: $\mathbf{T}$
  $m \leftarrow sizeof(\mathbf{T})$;
  for $i = 1 \dots m$ **do**
    $\bar{Y}(\mathbf{x}_i) =$NN$((\mathbf{T} \setminus \mathbf{x}_i), \mathbf{x}_i)$;
    $S \leftarrow$ Model$(\mathbf{T}, \mathbf{x}_i)$
    $\theta = \alpha \cdot std\left(Y\left(\mathbf{X}_S\right)\right)$
    if $\left|Y(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i)\right| > \theta$ **then**
      $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$
    end if
  end for
  $\mathbf{P} \leftarrow \mathbf{T}$
  return $\mathbf{P}$

Algorithm 2 RegCNN algorithm

Require: $\mathbf{T}$
  $m \leftarrow sizeof(\mathbf{T})$
  $\mathbf{P} = \emptyset$
  $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{x}_1$;
  for $i = 2 \dots m$ **do**
    $\bar{Y}(\mathbf{x}_i) =$NN$(\mathbf{P}, \mathbf{x}_i)$
    $S \leftarrow$ Model$(\mathbf{T}, \mathbf{x}_i)$
    $\theta = \alpha \cdot std\left(Y\left(\mathbf{X}_S\right)\right)$
    if $\left|Y(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i)\right| > \theta$ **then**
      $\mathbf{P} \leftarrow \mathbf{P} \cup \mathbf{x}_i$;
      $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$
    end if
  end for
  return $\mathbf{P}$

As the prediction algorithm for instance selection we use a weighted k-NN with $k = 9$, where the weight w_i exponentially decreases with the distance d_i between the given vector and its i-th neighbor x_i. The predicted output y is given by eq. 1.

$$y = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} \tag{1}$$

where $w_i = 2^{-0.2 d_i}$. We use Euclidean distance measure and a threshold Θ, which expresses the maximum difference between the output values of two vectors to consider them similar. Using Θ proportional to the standard deviation of k nearest neighbors of

the vector x_i reflects the speed of changes of the output around xi and allows adjusting the threshold to that local landscape, what, as the experiments showed, allows for higher compression of the dataset. As the regression model to predict the output Y(xi) we use k-NN with $k = 9$ as the Model(T,x_i) ($k = 9$ usually produced good results [15]). To adjust the CNN, ENN and CA algorithms to regression tasks the wrong/correct classification decision is replaced with a distance measure and a similarity threshold, to decide if a given vector can be considered as similar to its neighbors. Additionally in the CA algorithm, we define a regression counterpart of "the same class" by replacing it with "the output values being close enough". Thus, we have to find a pair of vectors, which are both: close in the input space (as in the original CA) and close in the output space. We merge these two vectors, for which the weighed sum D of the distances in the input and output space takes the smallest value:

$$D = d_{input} + \sqrt{num_{features}} \cdot d_{output} \tag{2}$$

The above formula requires the input and output values to be standardized. The distance in the output space is multiplied by the square root of the number of features to make the weight of it equal to the weight of the distance in the input space. If the priority is to obtain small dataset and thus make it easier to obtain simple logical rules from a regression tree built with the dataset, we can further modify the RegCA algorithm to allow for merging two closest vectors even if that slightly decreases the accuracy.

3 Regression Tree

A univariate regression tree is used with splitting of the data into nodes is based on variance minimization. The algorithms searches for all possible split points p of each feature f, as shown in the pseudo-code. The value optimized value is v and v_0 is the value at the previous iteration. Multiplying the variances of child nodes v_L, v_R by the number of vectors in the same nodes p_L, p_R causes that the splits are more symmetrical. Otherwise, it would be frequently only a few, or even one, vectors in one child node and all the other vectors in the other node. Such an unsymmetrical tree would have poor performance [16]. In our experiments the exponents n =1 and $m = 1$. To further improve the results, at the entrance of the tree the data can be transposed by a hyperbolic tangent to obtain rather uniform than Gaussian distribution [16]. One of the simplest forms of pruning is reduced error pruning. Starting at the leaves, each node is replaced with the average value of all vectors in the node and its subnodes. If the prediction accuracy does not decrease then the change is kept. There are two other possible ways to reduce the size of the tree. One of them is to use stopping criteria, such as minimal variance in the node. Once the minimal variance in the node is reached the node becomes a leaf. Another criterion that can be used together with the minimal variance is the minimal number of vectors in the node. If any node already reached that number of vectors if becomes a leaf. However, the problem with the stopping criteria is that their optimal value is unknown during building the tree and what is even more difficult, the optimal stopping criteria can be different for various nodes. Thus, the third discussed way to reduce the tree size is to reduce the training set size. If the training set is reduced with instance selection methods, the most representative instances remain in the

dataset, allowing for building a smaller tree with better prediction ability. In regression problems there are several parameters that can be tuned in the RegENN, RegCNN and RegCA algorithms. Depending on how the parameters are set more or fewer instances remain in the training set. If the parameters are set so that only few instances remain in the training set, then we can build the decision tree without any pruning or stopping criteria with a single instance in each leaf. Also any combinations are possible with either more intensive instance selection or more intensive pruning. The combinations are experimentally evaluated in the experimental section of the paper.

Algorithm 3 Tree optimization pseudo-code

Require: $\mathbf{F} = [f_1, f_2, \ldots, f_s]$
Ensure: $\underset{i=1:s}{\forall}$ sizeof$(f_i) \leftarrow p$
  for $i = 1 \ldots s$ **do**
    $f_i = SortFeatureElements(f_i)$
    for $j = 1 \ldots p$ **do**
      $p_L = j/p$
      $p_R = (p - j)/p$
      $v = v_0 - p_L^m \cdot v_L^n - p_R^m \cdot v_R^n$
      if $v \geq q$ **then**
        $q = v$
        $s_0 = j$
        $f_0 = f_i$
      end if
    end for
  end for
  return s_0, f_0

4 Numerical Experiments

We implemented the instance selection algorithms in Java as RapidMiner Extensions and used RapidMiner [17] for the whole process. The decision tree was created in C# as a WPF application with graphical visualization and it is invoked in the ExecuteProgram module (fig. 1.) as well in the training as in the test part of the validation process. All the source codes, executable files and datasets used in the experiments can be downloaded from [18]. The whole process in different configurations was run in a 10-fold crossvalidation loop. Inside each crossvalidation run first he instance selection was performed and then the regression tree was created on the training dataset. The MSE was measured on the test dataset. The Training and Testing (the two lower windows in fig.1) constitute together a single run of the 10-fold crossvalidation (represented by the Validation module in the upper window in fig. 1). In the testing part, first the instance selection is made in various ways (in fig. 1. RegENN followed by RegCA is shown) then the selected examples (prototypes) are written to a CSV file and the ExecuteProgram module run the regression tree in the training mode as the external program. The tree reads the CSV dataset and builds the tree. In the Testing block, the tree is invoked from the ExecuteProgram module and it reads the structure of the decision tree (which

was saved to a file in the Training block) and tests the tree on the test data, (the data was saved to a file by the WriteCSV(2) module). The ReadCSV(2) module reads the prediction results of the tree and the Performance module calculates the MSE on the data. The Log module in the MainProcess block calculates the MSE and its standard deviation over the whole crossvalidation.

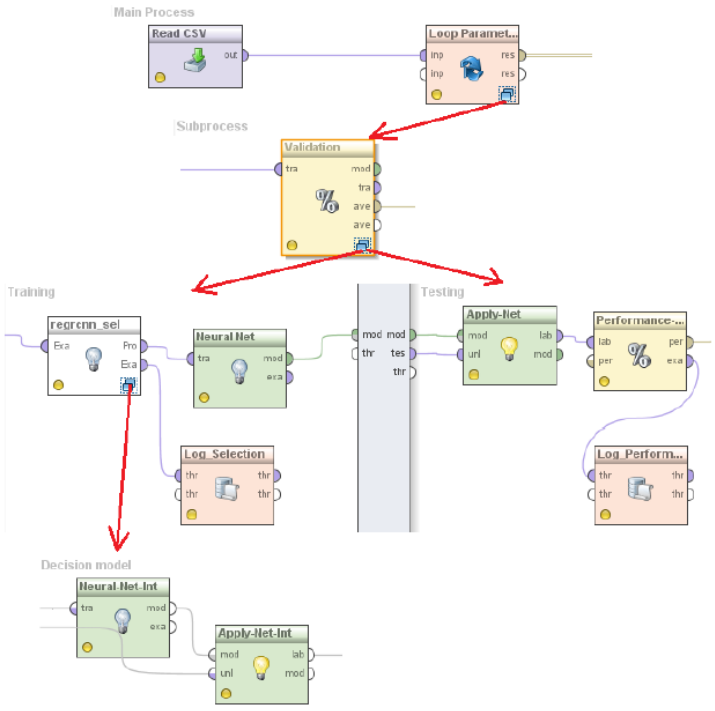

Fig. 1. The experimental process as implemented in Rapidminer

We performed the experiments on four datasets. First all the datasets were standardized so that the mean value of each attribute is zero and the standard deviation is one to make comparison of the results easy. Four datasets come the UCI Machine Learning Repository [19]: Concrete Compression Strength (7 attributes, 1030 instances), Crime and Communities (7 attr., 320 inst.), Housing (13 attr., 506 inst.). One dataset (Steel: 12 attr., 960 inst.) depicts the steel production process with the task to predict the amount of carbon that must be added to the liquid steel in to obtain desired steel properties. We experimentally evaluated the optimal Θ (see section IIB) and we used $\Theta = (5 \div 7) \cdot std_dev$ of 9 nearest neighbors for RegENN and $\Theta = (0.15 \div 0.25) \cdot std_dev$ of 9 nearest neighbors for RegCNN and RegCA. Additionally each time the node variance=0.001 was used as the stopping criteria.

Table 1. Experimental results for the Steel dataset.

Regularization method	MSE	nodes	vectors
node variance	0.23±0.07	77±10	864±0
tree pruning	0.20±0.07	62±8	864±0
ENN	0.23±0.07	70±8	776±2
CNN	0.20±0.06	68±8	746±8
CA	0.18±0.05	62±8	701±4
ENN+CNN	0.18±0.05	62±7	722±10
ENN+CA	0.17±0.04	56±6	687±5
ENN+CA+pruning	0.16±0.04	53±6	687±5

Table 2. Experimental results for the Crime dataset.

Regularization method	MSE	nodes	vectors
node variance	0.68±0.09	85±10	288±0
tree pruning	0.63±0.08	72±9	288±0
ENN	0.68±0.08	77±8	210±2
CNN	0.66±0.06	79±8	243±2
CA	0.63±0.06	72±8	202±3
ENN+CNN	0.64±0.07	72±9	197±4
ENN+CA	0.60±0.06	67±8	187±2
ENN+CA+pruning	0.58±0.06	63±8	187±2

Table 3. Experimental results for the Concrete dataset.

Regularization method	MSE	nodes	vectors
node variance	0.88±0.08	167±26	927±0
tree pruning	0.85±0.07	125±22	927±0
ENN	0.92±0.08	72±9	277±5
CNN	0.96±0.09	142±22	786±6
CA	0.88±0.07	122±18	609±5
ENN+CNN	0.86±0.07	63±6	186±4
ENN+CA	0.83±0.06	56±6	180±4
ENN+CA+pruning	0.82±0.06	48±6	180±4

Table 4. Experimental results for the Housing dataset.

Regularization method	MSE	nodes	vectors
node variance	0.41±0.08	105±14	455±0
tree pruning	0.39±0.08	68±11	455±0
ENN	0.43±0.09	70±10	350±4
CNN	0.39±0.08	68±10	387±3
CA	0.39±0.08	68±10	379±3
ENN+CNN	0.39±0.08	63±9	339±5
ENN+CA	0.36±0.06	63±9	319±4
ENN+CA+pruning	0.34±0.06	58±8	319±4

5 Conclusions

We presented an extension of CNN, ENN and CA algorithms, called RegCNN, RegENN and RegCA that can be applied to regression tasks and experimentally evaluated the influence of the Θ parameter on the number or selected vectors and the size and prediction accuracy of the regression tree. The best results were obtained when RegENN was used together with RegCA. It always improved the properties of the regression tree (smaller size and higher prediction accuracy), however after applying tree pruning in several cases the results could still be improved. Although the parameter Θ must be determined, the algorithms are not very sensitive to changes of the parameters and as rule of thumb it can be set to 6 for RegENN and to 0.2 for RegCNN and RegCA for standardized datasets. In some cases we were able to obtain slightly better results using more complex algorithms (e.g. and MLP network), which were trained only on a part of the dataset situated closest to the vector of interest (local experts). However, that increases the time of instance selection by two or three orders of magnitude and make the process more complex than the k-NN. So far we adjusted to regression only some of the simplest instance selection methods and it would be worth to perform the study with other methods as well. Although the instance selection in regression problems does not reduce the dataset size so much as in classification tasks, it is worth performing not only to compress the data but also to improve the prediction of the model.

Acknowledgment. The work was sponsored by the grant No. ATH/2/IV/GW/2011 from the University of Bielsko-Biala.

References

1. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14, 515-516 (1968)
2. Wilson, D.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics 2, 408-421 (1972)
3. Chang, C.L., Finding prototypes for nearest neighbor classifiers. IEEE Transactions on Computers 23, pp. 1179-1184 (1974)
4. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning, Vol. 38, 251-268 (2000)
5. Cameron-Jones, R.M., Instance selection by encoding length heuristic with random mutation hill climbing. The Eighth Australian Joint Conference on Artificial Intelligence. pp. 99-106 (1995)
6. Salvador, G., Derrac, J., Ramon, C.: Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34 417-435 (2012)
7. Jankowski, N., Grochowski, M.: Comparison of instances selection algorithms. LNCS, vol. 3070, 598-603 (2004)
8. Kordos, M., Blachnik, M., Wieczorek, T.: Temperature Prediction in Electric Arc Furnace ith Neural Network Tree. ICANN 2011, LNCS, vol. 6792, 71-78 (2011)
9. Zhang, J., et. al.: Intelligent selection of instances for prediction functions in lazy learning algorithms. Artifcial Intelligence Review, Vol. 11, 175-191 (1997)

10. Tolvi, J.: Genetic algorithms for outlier detection and variable selection in linear regression models. Soft Computing, Vol. 8, 527-533 (2004)
11. Guillen, A., et. al.: Applying Mutual Information for Prototype or Instance Selection in Regression Problems. ESANN'2009 proceedings (2009)
12. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for understanding of data. Proceedings of the IEEE, vol. 92, no. 5, pp. 771-805 (2008)
13. Shuning Wu: Optimal instance selection for decision tree, PhD dissertation. Iowa State university (2007)
14. Ramon Cano J., Herrera, F., Lozano, M.: Evolutionary Stratified Training Set Selection for Extracting Classification Rules with Trade off Precision-Interpretability. Data and Knowledge Engineering, Vol. 60, pp. 90-108 (2006)
15. Kordos, M., Strzempa, D., Blachnik, M.: Do We Need Whatever More than k-NN? ICAISC 2012, LNAI, vol. 6113, 414-421 (2010)
16. Kordos, M., et. al.: A Hybrid System with Regression Trees in Steel-making Process, LNCS, vol. 6678, 222-229 (2011)
17. www.rapid-i.com
18. www.mkordos.com/icann2013
19. Merz, C., Murphy, P.: UCI repository of machine learning databases www.ics.uci.edu/mlearn/MLRepository.html