Instance Selection with Neural Networks for Regression Problems

Mirosław Kordos[1], Marcin Blachnik[2]

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

Lecture Notes in Computer Science, vol. 7553, pp. 263-270, ICANN, September 2012
The original publication is available at www.springerlink.com

Abstract. The paper presents algorithms for instance selection for regression problems based upon the CNN and ENN solutions known for classification tasks. A comparative experimental study is performed on several datasets using multilayer perceptrons and k-NN algorithms with different parameters and their various combinations as the method the selection is based on. Also various similarity thresholds are tested. The obtained results are evaluated taking into account the size of the resulting data set and the regression accuracy obtained with multilayer perceptron as the predictive model and the final recommendation regarding instance selection for regression tasks is presented.

Keywords: neural network, instance selection, regression

1 Introduction

1.1 Motivation

There are two motivations for us to undertake research in the area of instance selection for regression problems. The first one is theoretical - most research on instance selection, which has been done so far refers to classification problems and there are only few papers on instance selection for regression tasks, which do not cover the topic thoroughly, especially in the case of practical application to real-world datasets. And our second motivation is very practical - we have implemented in industry several computational intelligence systems for technological process optimization [1], which deal with regression problems and huge datasets and there is a practical need to optimally reduce the number of instances in the datasets before building the prediction and rule models.

There are following reasons to reduce the number of instances in the training dataset:

1. Some instance selection algorithms, as ENN, which is discussed later, reduce noise in the dataset by eliminating outliers, thus improving the model performance.

2. Other instance selection algorithms, as CNN, which is also discussed later, discard from the dataset instances that are too similar to each other, what simplifies and reduces size of the data.
3. The above two selection models can be joined together to obtain the benefits of both.
4. The training performed on a smaller dataset is faster. Although reducing the dataset size also takes some time, it frequently can be done only once, before attempting various models with various parameters to match the best model for the problem.
5. While using lazy-learning algorithms, as k-NN, reducing the dataset size also reduces the prediction time.
6. Instance selection can be joined with prototype selection in prototype based systems, e.g. prototype rule-based systems.

1.2 Instance Selection in Classification Problems

In that area many research have been done using the k-nearest neighbor algorithm (k-NN) for instance selection in classification tasks. The early research in that area lead to the Condensed Nearest Neighbor rule (CNN) [2] and Edited Nearest Neighbor rule (ENN) [3]. These basic algorithms were further extended leading to more complex ones like Drop1-5 [6], IB3, Gabriel Editing (GE) and Relative Neighborhood Graph Editing (RNGE), Iterative Case Filtering (ICF), ENRBF2, ELH, ELGrow and Explore [7]. Instead of directly selecting instances from the training data an interesting approach for training k-NN classifier was proposed by Kunheva in [5], where preselected instances were relabeled, such that each instance was assigned to all class labels with appropriate weights describing the support for given label. A large survey including almost 70 different algorithms of instance selection for classification tasks can be found in [7]. The instance selection algorithms were designed to work with k-NN classifier, so an interesting idea was proposed by Jankowski and Grochowski in [4]; 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. In this paper we aim to obtain the same for regression tasks.

1.3 Challenges in Regression Tasks

The instance selection issue for regression tasks is much more complex. The reason is that in classification tasks only the boundaries between classes must be precisely determined, while in regression tasks the output value must be properly calculated at each point of the input space. Moreover, the decision in classification tasks is frequently binary or there are at most several different classes, while in regression tasks, the output of the system is continuous, so there are unlimited number of possible values predicted by the system. That causes that the dataset compression obtained by instance selection can be much higher in classification than in non-linear regression problems. Moreover, the decision about rejection of a given vector in classification tasks can be made based

on right or wrong classification of the vector by some algorithm. In regression problems, rather some threshold defining the difference between the predicted and the actual value of the vector output should be set. As discussed later, the threshold should be rather variable than constant, taking different values in different areas of the data. Thus, determining the threshold is an issue that does not exist in classification tasks. Another problem is the error measure, which in classification tasks is very straightforward, while in regression tasks, the error measure can be defined in several ways and in practical solutions not always the simple error definitions as the MSE (mean square error) work best [1].

Because of the challenges, there were very few approaches in the literature to instance selection for regression problems. Moreover, the approaches were usually not verified on real-world datasets. 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.

2 Methodology

2.1 ENN and CNN Instance Selection Methods

The CNN (Condensed Nearest Neighbor) algorithm was proposed by Hart [2]. For classification problems, as shown in [4], CNN condenses on the average the number of vectors three times. CNN used for classification works in the following way: 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. If the classification is correct, then the instance is not added to the final dataset P. If the classification is wrong - the instance is added to P. Thus, the purpose of CNN is to reject these instances, which do not bring any additional information into the classification process.

The ENN (Edited Nearest Neighbor) algorithm was created by Wilson [3]. The main idea of ENN is to remove given instance if its class is different than the majority class of its neighbors, thus ENN works as a noise filter. ENN starts from the entire original training set T. Each instance, which is correctly classified by its k nearest neighbors is added to the new dataset P and each instance wrongly classified is not added. Several variants of ENN exist. Repeated ENN, proposed by Wilson, where the process of ENN is iteratively repeated as long as there are any instances wrongly classified. In all k-NN algorithm proposed Tomek [12], the ENN is repeated for all k from k=1 to k_{max}.

2.2 RegENN and RegCNN: ENN and CNN for Regression Problems

The first step to modify the CNN and ENN algorithms to enable using them for regression task is to replace the wrong/correct classification decision with a distance measure and a similarity threshold, to decide if the examined vector can be considered as similar to its neighbors or not. For that purpose we use Euclidean measure and a threshold

θ, which express the maximum difference between the output values of two vectors to consider them similar. Using θ proportional to the standard deviation of several nearest neighbors of the vector $\mathbf{x}_i$ reflects the speed of changes of the output around $\mathbf{x}_i$ and allows adjusting the threshold to that local landscape, what, as the experiments showed, allows for better compression of the dataset. Then we changed the algorithm used to predict the output $Y(\mathbf{x}_i)$ from k-NN to an MLP (multilayer perceptron) neural network, which in many cases allowed for better results (see Table 1). Additionally the best results were obtained if the MLP network was trained not on the entire dataset, but only on a part of it in the area of the vector of interest. The algorithms are shown in the following pseudo-codes.

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 $\mathbf{P}$

Where $\mathbf{T}$ is the training dataset, $\mathbf{P}$ is the set of selected prototypes, $\mathbf{x}_i$ is the i-th vector, m is the number of vectors in the dataset ,$Y(\mathbf{x}_i)$ is the real output value of vector $\mathbf{x}_i$, $\bar{Y}(\mathbf{x}_i)$ is the predicted output of vector $\mathbf{x}_i$, S is the set of nearest neighbors of vector x_i, NN(A,x) is the algorithm, which is trained on dataset A and vector $\mathbf{x}$ is used as a test sample, for which the $Y(\mathbf{x}_i)$ is predicted (in our case NN(A,x) is implemented by k-NN or MLP), kNN is the k-NN algorithm returning the subset S of several closest neighbors to $\mathbf{x}_i$, and θ is the threshold of acceptance/rejection of the vector as a prototype, α is a certain coefficient (it will be discussed in the experimental section) and $std\left(Y\left(\mathbf{X}_S\right)\right)$ is the standard deviation of the outputs of the vectors in S.

2.3 Instance Selection Extension for RapidMiner

We used the RapidMiner [15] software for implementing the algorithms and performing numerical experiments. We created in Java the instance selection modules and incorporated them in the entire model. The source code of the instance selection modules, the .jar file containing them, that can be used with RapidMiner without the necessity to compile the sources, the .xml file with the process and the datasets we used can be obtained from [16]. The most outer loop of the model iterates over θ or α parameters, the lower level loop is a 10-fold crossvalidation, where in the training part the RegCNN

or RegENN algorithm is performed and an MLP network is trained on the selected instances, and in the test part this MLP networks is tested on the test set. The RegCNN or RegENN module contains the NN(A,x) algorithm which in Fig. 1 is an MLP network.

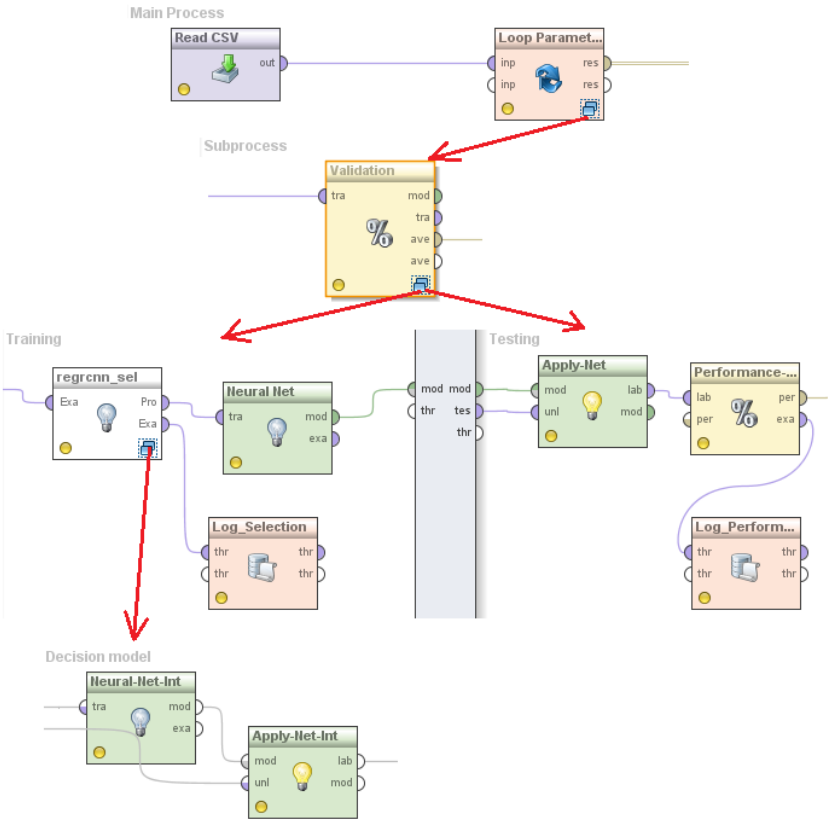

Fig. 1. The RapidMiner model used in the experiments

3 Experiments and Results

3.1 Dataset Description

We performed the experiments on five datasets. Before the experiments all the datasets were standardized so that the mean value of each attribute is zero and the standard deviation is one to enable easy comparison of the results obtained with different datasets and of the parameters used in the experiments. All the datasets in the form used in the experiments can be download from our web page [16]. These datasets included four datasets from the UCI Machine Learning repository [14]: Concrete Compression Strength (7 input attributes, 1030 instances), Forest Fires (10 input attributes, 518 instances), Crime

and Communities (7 input attributes, 320 instances), Housing (13 input attributes, 506 instances). One dataset (Steel: 12 input attributes, 960 instances) contains data from steel production process and the task here is to predict the amount of carbon that must be added to the liquid steel in the steel-making process to obtain steel of desired parameters, given the chemical and physical properties of the steel and the current melting process.

3.2 Experimental Results

Table 1. Results for Steel, Concrete, Crime, Forest and Housing datasets obtained with the optimal α parameters (where $\theta = \alpha \cdot std\left(Y\left(\mathbf{X}_S\right)\right)$

algorithm.		Steel	Concrete	Crime	Forest	Housing
ENN-3NN	MSE	0.23±0.08	0.94±0.10	0.69±0.09	1.44±0.3	0.46±0.11
	vect(α)	740±3(5)	275±6(4)	211±3(5)	373±6(1)	348±4(5)
ENN-9NN	MSE	0.25±0.08	0.92±0.08	0.68±0.09	1.42±0.3	0.43±0.10
	vect(α)	776±2(5)	277±5(4)	210±2(5)	379±5(10)	350±4(5)
ENN-MLP90	MSE	0.21±0.06	0.92±0.10	0.66±0.09	1.39±0.3	0.42±0.09
	vect(α)	772±3(5)	279±5(3.5)	206±1(6)	389±6(6)	365±3(6)
ENN-MLP30	MSE	0.21±0.06	0.90±0.10	0.67±0.08	1.39±0.3	0.41±0.08
	vect(α)	773±3(5)	245±6 (4)	209±1(8)	391±6(7)	359±3(6)
ENN-MLP10	MSE	0.23±0.06	0.90±0.10	0.67±0.08	1.38±0.3	0.42±0.08
	vect(α)	752±3(5)	210±5(3.4)	215±1(7)	388±7(10)	360±3(7)
CNN-3NN	MSE	0.24±0.06	0.97±0.10	0.68±0.07	1.41±0.3	0.40±0.09
	vect(α)	747±6(0.1)	782±7(0.4)	235±2(0.3)	429±6(0.1)	385±3(0.4)
CNN-9NN	MSE	0.23±0.07	0.96±0.10	0.66±0.06	1.38±0.3	0.39±0.08
	vect(α)	746±8(0.15)	783±6(0.4)	243±2(0.3)	429±6(0.8)	387±3(0.5)
CNN-MLP90	MSE	0.23±0.08	0.96±0.10	0.68±0.07	1.39±0.3	0.41±0.08
	vect(α)	741±6(0.1)	787±6(0.5)	240±2(0.3)	423±4(0.2)	387±3(0.3)
CNN-MLP30	MSE	0.21±0.07	0.95±0.09	0.67±0.07	1.38±0.3	0.42±0.10
	vect(α)	752±12(0.1)	784±7(0.5)	238±2(0.3)	426±4(0.1)	391±3(0.3)
CNN-MLP10	MSE	0.22±0.01	0.94±0.10	0.68±0.06	1.42±0.3	0.42±0.08
	vect(α)	763±10(0.1)	780±7(0.5)	243±2(0.3)	424±4(0.1)	387±3(0.3)
ENN-CNN	MSE	0.19±0.05	0.86±0.07	0.64±0.07	1.35±0.3	0.39±0.08
	vect	722±10	186±4	197±4	366±4	339±6
	α	5/0.1	3.5/0.5	8/0.3	8/0.5	6/0.3
No selection	MSE	0.23±0.08	1.01±0.10	0.71±0.10	1.50±0.3	0.40±0.08
	vec	864	927	288	466	455

We performed the experiments using the model described in the previous section. Both MLP networks (the network used for instance selection and the network used for the final prediction) had one hidden layer with sigmoid activation function and the number of hidden neurons was rounded to (number of attributes + 1) / 2 + 1. Both networks were trained with the backpropagation algorithm for 500 training cycles with learning rate=0.3 and momentum=0.2. The stopping error was 1.0E-5.

In the case of ENN, which is a kind of a noise filter, the θ should be rather big, because only the outliers should be rejected. In the experiments with ENN we used 50

different values of θ usually from 0.05 to 5.0 or α usually from 0.5(1) to 50(100) in the case of variable θ. On the other hand for the CNN algorithm θ should be small, because only if two vectors are very similar, one of them should be rejected. In the experiments with CNN we used 50 different values of θ from 0.001 to 0.5 or α usually from 0.01 to 5 for each dataset. We used the following algorithms to predict the value to the vector being selected:

- k-NN with k=3
- k-NN with k=9 (frequenty k close to 9 is the optimal value [13])
- MLP network trained on the entire training data within one validation of the cross-validation process (that is 90 percent of whole the dataset, therefore it is shown in the result section as CNN-MLP90 or ENN-MLP90)
- MLP network trained on 33 percent of the training vectors, which were closest to the considered vector (shown in the result section as CNN-MLP30 or ENN-MLP30)
- MLP network trained on 11 percent of the training vectors, which were closest to the considered vector (shown in the result section as CNN-MLP10 or ENN-MLP10)

Because of limited space, only the best results obtained for each model with variable θ are shown in the table and comparison of results obtained with various θ (constant and variable) are shown for one dataset and one method in a graphical form.

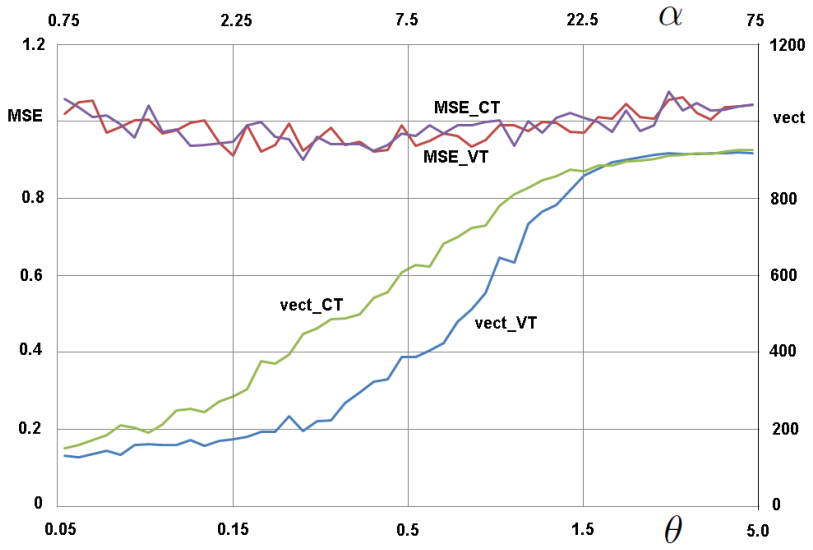

Fig. 2. Dependance of MSE (MSE_CT: with constant θ, MSE_VT: with variable θ) and the number of selected vectors (vect_CT: with constant θ, vect_VT: with variable θ) on the threshold θ (when it is constant) and on α (where$\theta = \alpha \cdot std\left(Y\left(\mathbf{X}_S\right)\right)$)

4 Conclusions

We presented an extension of CNN and ENN, called RegCNN and RegENN algorithms that can be applied to regression tasks and experimentally evaluated the influence of the θ and α parameter and various learning methods within the selection algorithm on the number or selected vectors and the prediction accuracy obtained with an MLP neural network on the reduced dataset. The general conclusions are that in most cases the best results are obtained using an MLP network trained on the subset of closest neighbors of the considered point. It was observer, that in general the θ used with CNN could be on average set to 0.1 (or α to 0.5) of the MSE value while performing prediction on the unreduced dataset, while the θ used with ENN to 5 times the value of the MSE (or α to 5) for standardized data, however, the algorithms are not very sensitive to the change of α in the terms of prediction accuracy, but especially RegENN with lower α allows for better dataset compression. Variable θ allows for reducing more vectors, while it does not influence the prediction accuracy. The best results are obtained if first ENN and after that CNN was applied to the dataset.

It should be possible to significantly improve the results, first by tuning the parameters of the MLP network and using more efficient MLP training methods, such as Levenberg-Marquardt algorithm and second by using more advanced instance selection methods, which were shortly presented in the introduction. These issues will be the area of our further research.

Acknowledgment. The work was sponsored by the grant ATH 2/IV/GW/2011 from the University of Bielsko-Biala.

References

1. Kordos, M., Blachnik, M., Wieczorek, T.: Temperature Prediction in Electric Arc Furnace with Neural Network Tree. Lecture Notes in Computer Science, vol. 6792, 71-78 (2011)
2. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14, 515–516 (1968)
3. Wilson, D.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics 2, 408–421 (1972)
4. Jankowski, N., Grochowski, M.: Comparison of instances selection algorithms. Lecture Notes in Computer Science, vol. 3070, 598-603 (2004)
5. Kuncheva, L., Bezdek, J.C.: Presupervised and postsupervised prototype classifier design. IEEE Transactions on Neural Networks, Vol. 10, 1142-1152 (1999)
6. D.R. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning, Vol. 38, 251-268 (2000)
7. 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)
8. Kovahi R., John, G.:Wrappers for Feature Subset Selection. AIJ special issue on relevance, May (1997)
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. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics 6, 448–452 (1976)
13. Kordos, M., Strzempa, D., Blachnik, M.: Do We Need Whatever More than k-NN? Lecture Notes in Artificial Intelligence, vol. 6113, 414-421 (2010)
14. Merz, C., Murphy, P.: UCI repository of machine learning databases http://www.ics.uci.edu/mlearn/MLRepository.html (1998-2012)
15. `www.rapid-i.com`
16. `www.kordos.com/icann2012`