Noise Elimination with a Re-Sampling Algorithm

H. Jair Escalante and Olac Fuentes

hugojair@ccc.inaoep.mx, fuentes@inaoep.mx
Department of Computer Science
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1
Santa María Tonantzintla, Puebla, México 72840

Abstract.
In this paper a new approach to noise detection and elimination in datasets for machine learning is presented. An algorithm that improves quality in training sets is introduced. This algorithm is based in the re-sampling idea that allows improving training data quality by identifying possible noisy instances and performing new measurements of each selected instance. We have obtained good results using this re-sampling algorithm in the prediction of stellar atmospheric parameters, a challenging astronomical domain where we tested the algorithm. We present experimental results of tests performed with varying noise levels that show how the re-sampling algorithm improves data quality, and hence classifier accuracy.

1. Introduction

In many machine learning applications the data to learn is imperfect, in real life domains the data available to train a classifier are not very reliable. Real datasets often include exceptional or noisy observations that degrade the induced classifier’s performance on unseen data. The exceptional observations are correct measurements that nevertheless their rare behavior may describe an unusual object. Noisy observations are incorrect measurements that can be due to several factors such as: human errors, instrument or measurement errors or bad experiment configuration. For this reason it is necessary to develop techniques that allow us to deal with these extraordinary observations.

Different approaches have been tried to solve this problem by using different techniques and methods to improve data quality. These approaches include the use of ensembles[4][5][15] to improve classifiers accuracy by filtering out noise data. In [5] they identify and then eliminate noisy instances by using majority vote and consensus filter ensembles. A similar approach is presented in [15] adding the use of bagging and boosting in an inductive logic programming domain. In [8] a comparison of four noise filtering algorithms is presented, including a coarse version of the work in [4]. The algorithms tested attained similar results in error, but differ in reduction rates. The use of the optimal margin classifier has been used to clean data [9]. Other
approaches do not perform a noise filtering method explicitly, but they leave out the instances that cause the highest errors in classifications. Such approaches perform instance selection trying to keep those instances that best approximate all possible cases of instances that could been present on unseen data. A survey on instance selection can be found in [3]. Also probabilistic approaches had been proposed, in [11] an expensive data cleaning method for two class problems is presented, that attempts to model the noise and the corruption process.

In this paper we propose a filtering technique based on re-sampling to improve data quality. Our goal is to reduce the noise in the training data, by performing several new measurements of the instances that seem to be noisy. In this approach none of the instances needs to be removed or weighted, but just re-sampled. In the next section we present the re-sampling algorithm and the fundamentals in which it is based. In Section 3 we describe the data and the astronomy problem, in Section 4 the experimental results are presented, and finally we present conclusions of this work and future work issues in Section 5.

2. Filtering the data

We have developed an algorithm that improves data quality based on the re-sampling idea, this algorithm is described in this section. The re-sampling algorithm is based on the observation that if we have more measurements of the same object then we can improve its quality. Most noise avoidance approaches perform instance elimination [4][5][8][15]. However not all suspicious instances are erroneous. In fact, if an instance seems suspicious it can be either a noisy or extraordinary, but correct, instance that represents a class. In this work we are not trying to make outlier detection or elimination explicitly. The algorithm is focused on noise elimination by performing several measurements of the possible noisy instances. For this reason this approach is oriented to those domains in which performing several measurements of the same object does not represents a high cost in resources.

2.1 The Re-Sampling Algorithm

The algorithm first identifies all possible noisy instances by means of a threshold approach. It consists of fixing a threshold error value, based on previous knowledge about the specific learning task, and to evaluate each training instance with a classifier. If an instance obtains an error value higher than our threshold the instance is labeled as “suspicious”. In our algorithm we perform a cross-validation approach to evaluate the algorithm performance.

Once we have identified all suspicious instances we proceed to apply the re-sampling technique. For each instance labeled as suspicious we perform k new measurements (1 ≤ k ≤ m) of the same object, where m is the maximum number of new measurements allowed, with the same configuration and in the same conditions that when were measured for the first time. We believe that if we perform several new measurements we can avoid the noise and only the useful information about an instance will remain.
This work then is oriented to those domains in which new measurements can do it in similar conditions that when the instance was obtained by the first time. Domains like: medicine, recognition of handwritten digits, astronomy and face recognition are well suited.

In each measurement the instance is evaluated using a machine learning algorithm and the training data available at that time. The stop criterion for the measurements is when an instance obtains low error levels based again on a previously fixed threshold or when m measurements are performed. If a suspicious instance has been measured more than m times and does not obtain low error levels, we substitute the attribute values of that instance by the average of the k measurements performed over that instance. On the one hand, if the instance labeled as suspicious is indeed noisy, then the re-sampling process will retain the useful information about that instance and the noise can be eliminated or reduced. On the other hand, if it is an extraordinary instance, then the values of that instance will remain almost intact, since the measurements’ average will remain very close to the value of the instance. Table 1 shows the re-sampling algorithm.

L is a machine learning algorithm which can be different in each step

1. Iterate n-times (n-fold cross validation)
   - Divide the dataset into T = training set and Q = test set
   - Build C the classifier using L and T, and predict the parameters for each instance in T with a n-fold cross validation. Obtain error values E(T) for each instance in T
   - Label as suspicious all instance in T such that: E(Ti) > error_threshold

Let j be the number of labeled instances

\[\text{confidence}_\text{level} := 1\]

- for i=1 to j  
  // re-sampling process

  while ( not low_error_flag or measurements_counter ≥ confidence_level)
    - Perform a measurement of the ith instance labeled and record its measurement value in measurements
    - Evaluate instance i with C

      if (error_ith < low_error_threshold )
        - low_error_flag := TRUE;
        - inc(measurements_counter);
      endwhile

    if (not low_error_flag)
      - instance i in T := average(measurements)
    else
      - instance i in T = actual_measurement
  • endfor

- Build C2 using T and L
- Use C2 to obtain error values of Q

2. end n-fold cross validation

Table 1. The re-sampling algorithm
As we can see the algorithm is very simple. In this case L can be the same machine learning algorithm in all cases, but is not a rule. The value of confidence_level indicates the maximum number of measurements allowed. error_threshold and low_error_threshold can have different values, according to our previous knowledge about the learning problem and the accuracy that we want to attain. In all the experiments we considered the mean absolute error of the prediction returned by LWR. For this work error_threshold as well as low_error_threshold were determined by analyzing the results attained in [7] [14], by performing several “trial and test” experiments over the dataset, and by previous knowledge about the specific domain. To find the adequate thresholds can be a very difficult problem, so we need to investigate another way to determine the suspicious instances in order to generalize the approach to other domains, this could be a direction for future work. low_error_threshold must vary with the different noise levels, since the permissible error of a 5% noise level is not the same that for 40% noise level.

In some cases an instance could not be improved, that is, after the k samples some instances do not obtain a error level lower than low_error_threshold even if we consider the average of the k samples. One way to deal with these instances is to pass them to an expert in the domain to determine if those instances are indeed noise or they are not. In this work only a few instances remain with errors higher than low_error_threshold after the improvement of the dataset, then the expert could easily determine the validity of the instances, this approach is similar to the work described in [9]. Another way is to vary L or the parameters of L, for example to consider more neighbors with a k-nearest neighbor or locally weighted linear regression approach or to train over more epochs and vary the number of hidden neurons if we were using neural networks.

2.2 Locally Weighted Linear Regression

In this work we used locally weighted linear regression (LWR) as our base machine learning algorithm. LWR is part of the group of instance based learning algorithms. This kind of algorithm stores some or all of the training examples and then postpone any generalization effort until a new instance’s class needs to be predicted. At this moment the algorithm attempt to fit the training examples only in a region around the query point.1

We choose LWR as our machine learning algorithm because its training time is equal to zero. We perform many experiments, so with LWR we avoid the training process time. Furthermore experiments performed and results from [7] shown that LWR outperforms artificial neural networks and self organizing maps in the prediction of stellar atmospheric parameters. The use of an ensemble could improves the accuracy [7], but the processing time will increases to much. In fact we could use LWR for the re-sampling phase and an ensemble for the validation of the test set.

There is no restriction in the choice of the machine learning algorithm, but it is very important to consider the characteristics of the problem approached.

1 For a detailed description about LWR see [1] and [12].
3. The Data

In this work we selected the problem of predicting stellar atmospheric parameters to test our re-sampling algorithm performance. We choose this regression task because the domain is adequate for the re-sampling algorithm operation. This paper explores an issue that had not been studied yet over this domain: noise elimination. Recent works have approached the estimation of these stellar parameters using different machine learning algorithms, such as artificial neural networks[2], genetic algorithms[13], instance-based learning[7][13] self organizing maps[7], ensembles[7] and other optimization algorithms[14]. Low error levels had been attained, however, noise elimination has not been approached directly.

![Sample stellar spectra](image)

3.1 Prediction of Stellar Atmospheric Parameters

A stellar spectrum is a plot of energy flux against wavelength, measured by a spectrograph, Figure 1 shows a stellar spectrum. As we can see, the plot presents several discontinuities called absorption lines. From the strength of the absorption lines an expert astronomer can estimate with good accuracy several of the most important properties of the star’s atmosphere including its temperature, surface gravity and metal content or metallicity. Instead of using the full spectra as input to our algorithm, a very large degree of compression can be attained without loss of accuracy [6] if we use a measurement of the strength of several absorption lines that are known to be important for predicting the stellar atmospheric parameters. In this work we use a library of such measurements called spectral indices, due to Jones [10]. This dataset consists of 48 spectral indices for 651 stars with their estimated temperature, surface gravity and metallicities. It was observed at the Kit Peak National Observatory.
3.2 Adding noise to the data

In this paper our principal objective is to filter noisy data and to evaluate the filtered training set on unseen data. We add random artificial noise with in different levels, there are many domains that.

By convenience we normalized the spectral indices to the range [0,1]. The way we insert noise is adding the generated noise to each of the stellar spectral indices and therefore all data are noise affected. Then when we say that we applied noise of 10% we add random normally distributed numbers between [-0.1,0.1] to all stellar spectral indices, for 20% between [-0.2, 0.2] and so on. Contrary to [15] where they apply noise only to a percentage of selected instances. Also our approach is different from [4],[5] in which a selective and more realistic noise introduction is presented.

By our definition of the noise applied to the spectral indices, if we perform various measurements on the same instance the noise will tend toward zero, and therefore the information obtained from the average of the measurements performed will contain the spectral indices with low noise level. In Figure 2 we can see an observation from the original dataset and the same observation affected with noise. As we can see there is a notable difference between the two instances. The re-sampling approach works well with the type of noise added. In theory human errors, bad experiment configuration or instrument and measurement errors could be avoided if in each sample the user ensures that the instruments and experiments configuration are the correct. Then the noise added by these means will be eliminated, independently of the domain approached. Mislabeled instances can not been improved, but could be eliminated in a preprocessing step.

![Fig. 2. Normalized stellar spectral indices from the original dataset (left) and affected with 40% of noise(right).](image)

4. Experimental Results

In this section we will show experimental results of our algorithm in the prediction of the stellar atmospheric parameters: temperature (Temp.), surface gravity (Grav.) and metallicity (Met.). Each result in the next tables is the average of three
experiments performed. The noise levels are 5, 10, 20 and 40 %. We fixed the confidence level value to 3 and 5, that is we performed 3 or 5 new measurements of the same instance. In all the experiments we used LWR as our base machine learning algorithm. The percentages presented in each table corresponds to the error reduction, taking as base the performance of LWR.

In table 2 we present the performance of our algorithm inside the 10-fold cross validation loop. In this case LWR were tested with 10-fold cross validation too, considering the same data. Table 2 shows error reductions ranging from 8 % to 32 %. We can observe that performing 5 measurements yields in better results than when we perform only 3.

<table>
<thead>
<tr>
<th>Confidence Level</th>
<th>Noise %</th>
<th>% Temp.</th>
<th>% Grav.</th>
<th>% Met.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5%</td>
<td>7.86</td>
<td>1.76</td>
<td>14.23</td>
</tr>
<tr>
<td>5</td>
<td>5%</td>
<td>12.49</td>
<td>11.63</td>
<td>18.11</td>
</tr>
<tr>
<td>3</td>
<td>10%</td>
<td>11.31</td>
<td>13.75</td>
<td>17.20</td>
</tr>
<tr>
<td>5</td>
<td>10%</td>
<td>17.78</td>
<td>15.63</td>
<td>18.37</td>
</tr>
<tr>
<td>3</td>
<td>20%</td>
<td>19.50</td>
<td>18.30</td>
<td>19.44</td>
</tr>
<tr>
<td>5</td>
<td>20%</td>
<td>26.17</td>
<td>21.02</td>
<td>15.29</td>
</tr>
<tr>
<td>3</td>
<td>40%</td>
<td>25.33</td>
<td>9.00</td>
<td>8.51</td>
</tr>
<tr>
<td>5</td>
<td>40%</td>
<td>32.46</td>
<td>15.63</td>
<td>15.24</td>
</tr>
</tbody>
</table>

Table 2. Results of the experiments performed on the filtering of data. Percentages show the error reduction in percentage compared with the performance of LWR.

In order to test that the re-sampling algorithm really improves data quality we performed experiments using the improved dataset to predict the parameters for both unseen noise-free data and unseen noisy data.

In table 3 we show the results of using the improved data with different noise levels as training set and then to predict 100 unseen noise-free instances. We can observe how the maximum reduction is attained when the training data contained 10% of noise. The minimum reduction were obtained when the data contained 40% of noise this due to that with 40% of noise our algorithm requires to perform more than 5 measurements in order to eliminate the noise from the spectral indices.

Also from table 3 we can see that performing 3 measurements give us better results than performing 5, column 3 rows 5 to 8. These results are interesting since they do not follow the reduction error pattern from table 2. We believe that this is due to the level and the randomness of the noise added (20% and 40%). Furthermore in that experiments the dataset was better improved performing 3 re-samplings instead 5, that is, 3 new samples or less were enough to improve the dataset. On the other hand, 5 re-samplings were not enough to improve the dataset.

On table 4 we observe results of using the improved dataset to train our algorithm and test on unseen noisy data. Is compared with LWR using as training set a set of data with different noise levels and then test with unseen data affected with the same noise levels. On the one hand, we can observe how our algorithm does not perform well on unseen noisy data, this due to that our algorithm improves the dataset
eliminating noise, then when is tested with noisy data it has a poorly performance. On the other hand, LWR learns (Overfit) the noise and then performs well on unseen noisy data. Regardless of the bad operation of our algorithm on unseen noise-free data, the algorithm performs well its main task: improves the dataset quality by noise elimination.

Table 3. Performance of the improved training dataset used to predict unseen stellar spectral indices without noise.

<table>
<thead>
<tr>
<th>Confidence Level</th>
<th>Noise %</th>
<th>% Temp.</th>
<th>% Grav.</th>
<th>% Mct.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5%</td>
<td>13.31</td>
<td>7.11</td>
<td>16.19</td>
</tr>
<tr>
<td>5</td>
<td>5%</td>
<td>25.17</td>
<td>26.90</td>
<td>24.55</td>
</tr>
<tr>
<td>3</td>
<td>10%</td>
<td>21.59</td>
<td>22.11</td>
<td>33.36</td>
</tr>
<tr>
<td>5</td>
<td>10%</td>
<td>35.01</td>
<td>36.45</td>
<td>37.70</td>
</tr>
<tr>
<td>3</td>
<td>20%</td>
<td>30.12</td>
<td>31.86</td>
<td>28.33</td>
</tr>
<tr>
<td>5</td>
<td>20%</td>
<td>21.80</td>
<td>31.83</td>
<td>42.62</td>
</tr>
<tr>
<td>3</td>
<td>40%</td>
<td>27.06</td>
<td>12.37</td>
<td>0.33</td>
</tr>
<tr>
<td>5</td>
<td>40%</td>
<td>23.09</td>
<td>16.39</td>
<td>6.80</td>
</tr>
</tbody>
</table>

Table 4. Anti-reduction of our algorithm compared with LWR performance. As training set we use our improved noise-free data set and we predict stellar parameters for unseen noisy-data.

<table>
<thead>
<tr>
<th>Confidence Level</th>
<th>Noise %</th>
<th>% Temp.</th>
<th>% Grav.</th>
<th>% Mct.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5%</td>
<td>-13.80</td>
<td>-6.25</td>
<td>-21.55</td>
</tr>
<tr>
<td>5</td>
<td>5%</td>
<td>-42.63</td>
<td>-22.62</td>
<td>-27.72</td>
</tr>
<tr>
<td>3</td>
<td>10%</td>
<td>-14.54</td>
<td>-5.98</td>
<td>-22.93</td>
</tr>
<tr>
<td>5</td>
<td>10%</td>
<td>-29.00</td>
<td>-13.67</td>
<td>-46.26</td>
</tr>
<tr>
<td>3</td>
<td>20%</td>
<td>-27.43</td>
<td>-13.45</td>
<td>-42.34</td>
</tr>
<tr>
<td>5</td>
<td>20%</td>
<td>-27.87</td>
<td>-20.42</td>
<td>-43.32</td>
</tr>
<tr>
<td>3</td>
<td>40%</td>
<td>-11.63</td>
<td>-30.77</td>
<td>-26.06</td>
</tr>
<tr>
<td>5</td>
<td>40%</td>
<td>-12.89</td>
<td>-48.97</td>
<td>-53.92</td>
</tr>
</tbody>
</table>

Finally we performed experiments using the improved training set to predict the parameters for 100 test instances with different noise levels but performing new measurements of each test instance indiscriminately. Results of this experiments are shown in table 5. As we can see, the results are evidence that the re-sampling technique is a good method for improve data quality.
Table 5. Reduction of our algorithm compared with LWR. As training set we use our improved data set and we predict stellar parameters for unseen noisy-data, performing several samplings of each test instance.

5. Conclusions and Future Work

We presented a re-sampling algorithm that filters noisy data by performing new measurements of the same instances. We tested our algorithm in the prediction of stellar atmospheric parameters. The results obtained showed that the algorithm improves data quality and therefore classifier accuracy. Experiments testing the improved training set using both noise-free test data and noisy test data were performed. Our algorithm outperformed LWR when it was tested with noise-free data, but performs poorly when it was tested with noisy data. This is due to the improvement of the training data, that is, the noise is eliminated from the data, then when the improved training data is used to approximate noisy data LWR performs poorly. However the objective of filtering noisy data is accomplished. Future work includes revision and development of techniques that allow us to use the re-sampling algorithm on data, when we have information only about the instance’s attributes. Also experiments on different domains need to be performed in order to test our algorithm in other situations.

References


