

# Data Cleaning Algorithms with Applications to Micro-Array experiments

Jason Weston, Olivier Chapelle, Isabelle Guyon  
BIOwulf Tech / BHT Labs  
*jason@barnhilltechnologies.com*  
*ochapell@ens-lyon.fr*  
*isabelle@clopinet.com*

April 7, 2001

## Abstract

In this note we make a first analysis of data cleaning methods for micro-array data. Experiments are performed on the colon cancer dataset from [1], the leukemia dataset of [4]. The final conclusion is that: sub-sampling is the method of choice in terms of prediction error (of mislabeled data) and that it also has the advantage of a naturally interpretable output which is useful for analysis of data anyway. We therefore decided to include it in the micro-array code library as the method of choice. We believe from experiments with oracle algorithms (with knowledge about the true labels or the true decision rule) that it is not possible to improve these results a lot if you stay within the framework of supervised learning.

## 1 Introduction

In this note we describe some investigations into the effectiveness of basic data cleaning techniques for micro-array data. We base the algorithms we test on the papers of [5], [7] and [3]. In previous work only Furey et al. applied their approach to Micro-Array data, however we perform a more thorough study. Supplementary to these algorithms, we also suggest new algorithms including a sub-sampling approach (in the work by Furey et al. leave-one-out is used, however we show that sub-sampling approaches can provide better results.)

Data cleaning is the problem of identifying mislabeled or meaningless data points. During data collection, several kinds of errors can be introduced: e.g hardware failures can cause the insertion of meaningless patterns or human failures can cause the insertion of mislabeled patterns. Removal of these patterns (or correction of *mislabeled*) might improve the performance of the classifier, and in the case of feature selection algorithms leads to more meaningful features. We will focus on data cleaning applications in biclass pattern recognition problems, although the reasoning should also apply to other domains.

It is also important to stress that not all outliers are mislabeled or meaningless patterns. They can be well labeled ambiguous patterns or atypical very informative patterns. This is what makes the tasks challenging and sheds doubt on the possibility of doing “automatic” cleaning. As we will show in experiments, manual cleaning can improve performance substantially (by removing patterns known by an oracle to be bad). However even automatic cleaning yields improvements (not large in our experiments, but in some experiments by Guyon et al. on character recognition [7], automatic cleaning was almost as good as manual cleaning, which is quite surprising). Moreover, even removing well labeled examples can improve performance. We believe these issues contribute to the non-triviality of the problem.

In pattern recognition we are usually given training data  $\mathbf{x}_i \in \mathbb{R}^n, i = 1, \dots, \ell$  with corresponding function values  $y_i \in \{\pm 1\}, i = 1 \dots, \ell$ . The vectors  $\mathbf{x}_i$  can be partially corrupted with noise, and thus still be at least partially useful. In this case the learning algorithm should take this into account. Many learning algorithms are designed to deal with this problem. Often algorithms assume a model of the noise, e.g the data and the noise are drawn from two probability distributions  $p(\mathbf{x}, y)$  and  $\xi(\mathbf{x})$  to give training data of the form  $\mathbf{x}_i + \xi_i, i = 1, \dots, \ell$ .

In the problem of data cleaning one is interested in identifying outliers: the situation where some of the data is completely meaningless or mislabeled, which does not easily fit into the usual model described above. Again, the problem of mislabeled data is also similar to the problem of corruption of the data with label noise but again, one cannot assume a model of the noise. We would like to identify these data points and remove them from the database. In the next section we try to formally define the problems we are trying to solve.

## 2 The Problem of Data Cleaning

The data cleaning problem can be understood in at least two different ways: the problem of data cleaning with mislabeled examples and the problem of minimizing prediction error (or example selection):

**Identifying outliers:** Training data  $(\mathbf{x}_i, y_i)_{i=1, \dots, \ell}$  are drawn from the true distribution  $p(\mathbf{x}, y)$ . However, instead of the labels  $y_{i=1, \dots, \ell}$  one is given the noisy labels  $y_1^*, \dots, y_\ell^*$  where  $n$  data points are mislabeled so that  $y_{p_i}^* = -y_{p_i}$ ,  $i = 1, \dots, n$  where  $p$  are indexes to the mislabeled patterns and  $0 \leq n \leq \ell$  (it is possible that no points are mislabeled.) Additionally, a further  $m$  patterns with indexes  $q_1, \dots, q_m$  are meaningless and have effectively a random  $\mathbf{x}$  vector (they are outliers). The task is to find the indexes  $p$  and  $q$  of the mislabeled or meaningless patterns.

In this report we will focus only on identifying mislabeled patterns.

**Minimizing prediction error:**<sup>1</sup> One would like to remove  $n$  examples

---

<sup>1</sup>One could also imagine a version of this problem where one considered flipping the labels of data points in order to find the best prediction error under the assumption that some of the examples have been mislabeled.

with indexes  $p_{i=1,\dots,n}$  from the learning set  $S_\ell = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$  in order to minimize the prediction error:

$$R(p) = \int \frac{1}{2} |y - f(\mathbf{x} | S_\ell \setminus (\mathbf{x}_i, y_i)_{i=p_1, \dots, p_n})| p(\mathbf{x}, y) \quad (1)$$

where  $f(\mathbf{x}, S)$  is the estimate of the label of point  $\mathbf{x}$  given by a learning algorithm (chosen *a priori*) trained on data set  $S$ .

These two descriptions also correspond to two possible courses of action. In the first one might wish to refer the potentially mislabeled or meaningless examples to the labeler (for example, the specialist who created the data, or someone further along the line of data collection and assembly) and query if this point is in fact mislabeled. (Dr. Guyon did exactly this with the prostate cancer data from Dr. Stamey and he came back to her with the finding that indeed the data was mislabeled). The ultimate aim of this is then either to correct mislabelings or to increase generalization ability of the learning algorithm.

The second course of action is *automatic data cleaning* where the learning algorithm chooses which data points are causing generalization ability to decrease and removes them without intervention from the user. This can be seen as a “dual” to the *feature selection problem* where one wishes to remove “bad features” not “bad examples”. We thus will also refer to the problem of data cleaning as *example selection*. Note that generalization ability can be improved by removing examples which are not necessarily mislabeled or meaningless.

In the spirit of usual solutions to the feature selection problem one can reduce both problems to two separate subproblems:

1. ranking the data points according to the likelihood that they should be removed; and
2. selecting how many of the patterns should be removed

The second problem can be seen as analogous to model selection. In this article we will only consider the problem of identifying incorrect patterns and the subproblem of ranking the data points.

In the next Section we will describe several algorithms to solve this problem and in Section 3 we compare these algorithms experimentally on toy data and real life Micro-Array problems. Section 4 then draws conclusions, and suggests further topics of research.

### 3 Experimental setup

In particular, we would like to test the ability of algorithms to mislabeled patterns. To this end our experimental setting is to deliberately mislabel training points and test the ability of algorithms to detect this mislabeling.

This mislabeling process could include many mislabeled points but in our experiments we only mislabel a single point at a time.

**algorithms** All of the algorithms that we test are of the following form: they are given as input a training set  $S_\ell$  and they output a rank  $r_i$  for the  $\ell$  data points. The rank indicates the likelihood of being a mislabeled pattern, with greater likelihood being assigned a higher rank.

**scoring algorithms** Measuring the success of a data cleaning algorithm even when there is only a single mislabeled point is not obvious. Taking the mean rank of the (single) mislabeled points is an obvious measure, however it can pay too much attention to a few large scores. For example, if one algorithm predicts the mislabeled points on 9 out of 10 runs with rank 1 but on the other run it assigns a rank of 50 it obtains a mean of 5.9. If another algorithm always gives a rank of 5 it has a mean of 5 however we think the first algorithm is more useful for detecting outliers. If one is referring the first one or two points to the data collector for verification then the first algorithm will find outliers whereas the second will not. We therefore record the following measurements of error over  $n$  runs:

- number of mislabeled points with rank 1,2 and 3 (three separate scores)
- mean rank of the mislabeled points:  $\frac{1}{n} \sum_i r_{p_i}$  where  $p_i$  is the index of the mislabeled point in run  $i$ .
- “trimmed” mean:  $\frac{1}{n} \sum_i \min(c, r_{p_i})$  where  $p_i$  is the index of the mislabeled point in run  $i$ . That is, we take the mean rank of the mislabeled points, but for ranks greater than  $c$  we decrease their contribution to the mean. By doing this, we attempt to concentrate the score on the first few rankings. We use  $c = 5$  and  $c = 10$  and in tables of results we refer to these scores as “mean  $\leq 5$ ” and “mean  $\leq 10$ ” respectively.

**data generation** Our setup is therefore as follows. In toy data we randomly draw 100 training sets of a fixed size and then flip the label of a single training example in each set. We then score the data cleaning algorithms on each dataset according to the ranking of the mislabeled example in each set, and take the mean score. In real datasets that are of a small size (e.g microarray data) we cannot randomly draw many independent training sets. We therefore use the following procedure: we flip the label of each example in turn so that one has  $\ell$  copies of the original training set but each with a single mislabeled point. One then scores the data cleaning algorithms according to the ranking of the mislabeled example in each set, and takes the mean score.

## 4 Algorithms for Data Cleaning

We compare the following algorithms:

**SVM- $\xi$**  This algorithm records the distance from the “correct” side of the margin ( $\xi_i$ ) of each training point. The ranking is thus given by the distance of each point from the “correct” side of the margin, largest ranked first.

- Choose a value of the soft margin parameter  $C$ .
- Train the classifier  $[w, b] = \text{SVM}(S_\ell, C)$ .
- Calculate  $\xi_i = 1 - y_i(w \cdot \mathbf{x}_i + b)$  for all  $i$ .
- Assign  $r_i = \text{card}\{\xi_j : \xi_j \geq \xi_i\}$  for all  $i$ .

**SVM- $\alpha$**  This algorithm records the size of the weight ( $\alpha_i$ ) of each training point. If a point is easy to classify it has weight zero, the more “unusual” the point, the larger the weight. The ranking is thus given by the weight of each point, largest ranked first.

- Choose a value of the soft margin parameter  $C$ .
- Train the classifier  $[\alpha, b] = \text{SVM-DUAL}(S_\ell, C)$ .
- Assign  $r_i = \text{card}\{\alpha_j : \alpha_j \geq \alpha_i\}$  for all  $i$ .

**SUB-ERR** This algorithm sub-samples the data many times, each time training an SVM and recording if each test point is mislabeled by the algorithm or not. The ranking is given by the average number of mislabelings, most mislabeled ranked first.

- Choose a value of the soft margin parameter  $C$ , the number of sub-sample runs  $p$  and the sub-sampling size  $q$ .
- Initialize  $e_i = 0$  and  $u_i = 0$  for all  $i$ .
- FOR  $i=1$  TO  $p$  runs
  - Draw a random sub-sample of the training data of size  $q$  with indexes  $tr_{j=1, \dots, q}$ .
  - Let the remainder of the data have indexes  $tst_{j=1, \dots, \ell-q}$ .
  - Train a classifier  $[w, b] = \text{SVM}(\{(\mathbf{x}_j, y_j)\}_{j=tr_1, \dots, tr_q}, C)$ .
  - Assign  $u_{tst_j} = u_{tst_j} + 1$  for all  $j$ .
  - Assign  $e_{tst_j} = e_{tst_j} + \frac{1}{2}|y_{tst_j} - \text{sign}(w \cdot \mathbf{x}_{tst_j} + b)|$  for all  $j$ .
- Assign  $r_i = \text{card}\{e_j/u_j : e_j/u_j \geq e_i/u_i\}$  for all  $i$ .

**SUB- $\alpha$**  This algorithm sub-samples the data many times, each time training an SVM and recording the weight ( $\alpha$ ) of each training point. If a point is easy to classify it has weight zero, the more “unusual” the point, the larger the weight. The ranking is thus given by the average weight of each point, largest ranked first.

- Choose a value of the soft margin parameter  $C$ , the number of sub-sample runs  $p$  and the sub-sampling size  $q$ .
- Initialize  $e_i = 0$  and  $u_i = 0$  for all  $i$ .
- FOR  $i=1$  TO  $p$  runs
  - Draw a random sub-sample of the training data of size  $q$  with indexes  $tr_{j=1,\dots,q}$ .
  - Let the remainder of the data have indexes  $tst_{j=1,\dots,\ell-q}$ .
  - Train a classifier  $[\alpha, b] = \text{SVM-DUAL}(\{(\mathbf{x}_j, y_j)\}_{j=tr_1,\dots,tr_q}, C)$ .
  - Assign  $u_{tr_j} = u_{tr_j} + 1$  for all  $j$ .
  - Assign  $e_{tr_j} = e_{tr_j} + \alpha_j$  for all  $j$ .
- Assign  $r_i = \text{card}\{e_j/u_j : e_j/u_j \geq e_i/u_i\}$  for all  $i$ .

**SUB- $\xi$**  This algorithm sub-samples the data many times, each time training an SVM and recording the average distance from the “correct” side of the margin ( $\xi_i$ ) of each training point. The ranking is thus given by the average distance of each point from the “correct” side of the margin, largest ranked first.

- Choose a value of the soft margin parameter  $C$ , the number of sub-sample runs  $p$  and the sub-sampling size  $q$ .
- Initialize  $e_i = 0$  and  $u_i = 0$  for all  $i$ .
- FOR  $i=1$  TO  $p$  runs
  - Draw a random sub-sample of the training data of size  $q$  with indexes  $tr_{j=1,\dots,q}$ .
  - Let the remainder of the data have indexes  $tst_{j=1,\dots,\ell-q}$ .
  - Train a classifier  $[w, b] = \text{SVM}(\{(\mathbf{x}_j, y_j)\}_{j=tr_1,\dots,tr_q}, C)$ .
  - Assign  $u_{tst_j} = u_{tst_j} + 1$  for all  $j$ .
  - Assign  $e_{tst_j} = e_{tst_j} + 1 - y_{tst_j}(w \cdot \mathbf{x}_{tst_j} + b)$  for all  $j$ .
- Assign  $r_i = \text{card}\{e_j/u_j : e_j/u_j \geq e_i/u_i\}$  for all  $i$ .

**LOO- $\xi$**  This algorithm performs leave-one-out, each time training an SVM and recording the distance from the “correct” side of the margin ( $\xi_i$ ) of the left out training point. The ranking is thus given by the distance of each left out point from the “correct” side of the margin, largest ranked first.

- Choose a value of the soft margin parameter  $C$ .
- FOR  $i=1$  TO  $\ell$ 
  - Train the classifier  $[w, b] = \text{SVM}(S_\ell \setminus \{(\mathbf{x}_i, y_i)\}, C)$ .
  - Assign  $e_i = 1 - y_i(w \cdot \mathbf{x}_i + b)$ .
- Assign  $r_i = \text{card}\{e_j : e_j \geq e_i\}$  for all  $i$ .

**LOO- $W^2$**  This algorithm performs leave-one-out, each time training an SVM and recording the size of the margin (which is inversely proportional to  $W^2(\alpha)$  (cf. equation 2) ). The ranking is thus given by the size of  $W^2$  when leaving out each point, largest ranked first.

- Choose a value of the soft margin parameter  $C$ .
- FOR  $i=1$  TO  $\ell$ 
  - Train the classifier  $[\alpha, b] = \text{SVM-DUAL}(S_\ell \setminus \{(\mathbf{x}_i, y_i)\}, C)$ .
  - Assign  $e_i = W^2(\alpha)$ .
- Assign  $r_i = \text{card}\{e_j : e_j \leq e_i\}$  for all  $i$ .

**FLIP- $W^2$**  This algorithm flips the label of each example, each time training an SVM and recording the size of the margin (which is proportional to  $W^2(\alpha)$ ). The ranking is thus given by the size of  $W^2$  when flipping each point, largest ranked first.

- Choose a value of the soft margin parameter  $C$ .
- FOR  $i=1$  TO  $\ell$ 
  - Train the classifier  $[\alpha, b] = \text{SVM-DUAL}(\{S_\ell \setminus \{(\mathbf{x}_i, y_i)\}\} \cup \{(\mathbf{x}_i, -y_i)\}, C)$ .
  - Assign  $e_i = W^2(\alpha)$ .
- Assign  $r_i = \text{card}\{e_j : e_j \leq e_i\}$  for all  $i$ .

There are also three other variants of this algorithm: **FLIP-SPAN** which uses the span [2] as the quality measure, **FLIP-DIST** which uses the change in distance from the margin of the flipped point and **FLIP-VALID** which uses a validation set instead, summing the errors using a sigmoid on the distance from the margin of type  $1/(1 + \exp(3x))$ .

**GRAD- $C$**  This algorithm tries to learn which data point is an outlier by assigning a variable  $C_i$  for each training point and minimizes  $R^2W^2$  using the ridge “trick” [2]. That is it minimizes

$$\sup_{\alpha} R^2(C)W^2(\alpha, C)$$

where

$$W^2(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j + \frac{1}{C_i} \delta_{ij}). \quad (2)$$

subject to

$$0 \leq \alpha_i, \quad \sum_i \alpha_i y_i = 0$$

and

$$R(C) = \frac{1}{\ell} \sum_i (x_i^2 + \frac{1}{C_i}) - \frac{1}{\ell^2} \sum_{i,j} [(x_i \cdot x_j) + \frac{1}{C_i} \delta_{ij}]$$

which can be solved by gradient descent.

One then assigns  $r_i = \text{card}\{C_j : C_j \leq C_i\}$  for all  $i$ .

**GOD** Finally, we also compared the results to three oracle algorithms which incorporate knowledge about the true decision function or true labels. These are: **SVM-GOD** which returns the largest  $\xi_i$  from a hard margin SVM trained on a second training set the same size as the original but containing only true labels (this is a different set, not just the same set with corrected labels, only the size of the sets is equal). **SVM-GOD  $C^*$**  is the same but with a soft margin. Finally, **BAYES-GOD** takes the furthest point on the “wrong” side of the optimal Bayes decision.

## 4.1 Feature Selection

One can also improve the data cleaning algorithms by using feature selection techniques. For example if one is analysing micro-array data where feature selection techniques are known to improve prediction performance then these techniques would probably also make it easier to detect outliers. One scheme is just to implement the feature selection algorithm directly into the classifiers. A second scheme is to consider different feature selection subset sizes and consider the scores across all the subset sizes (e.g a kind of averaging scheme). A similar scheme to this is described in [3].

However, care should be taken. When one of the authors (Isabelle Guyon) found a very severe outlier in the prostate cancer data, it was misclassified by leave-one-out for all gene subsets selected. Other patterns were misclassified only for some subsets. If one uses feature selection as part of cleaning, we would

suggest to put a higher weight on the results for large feature subsets since the feature selection mechanism tries to find the features that accomodate all the examples, even the bad ones.

In this article, as we are already considering a number of possibilities we make the assumption in this article that all the algorithms would be improved in the same way were we to use the same feature selection technique on all of them, and therefore we compare the algorithms without feature selection.

## 5 Experiments

### 5.1 Toy data

We generated a toy problem to first test the methods. The problem consists of two Gaussian distributions in  $d = 100$  dimensions with variance  $1.5/\text{sqrt}(d)$ , one for each class label, drawing 30 training points randomly for each class. We generated 100 of such datasets and flipped the label of the 30<sup>th</sup> data point (to help prevent bias in the sorting algorithm if there are equal ranks). The problem is then to detect that this data point is mislabeled. We compared all the algorithms described in the previous section in both hard margin and soft margin settings. The results are shown in Figure 1 and in more detail (but hence more difficult to parse) in Figure 2.

From these experiments we made the following conclusions:

- SVM- $\alpha$  is not one of the best algorithms to use
- SVM- $\xi$  is simple and obvious but a reasonable algorithm, however one needs to choose a good value of  $C$ .
- Sub-sampling is in general very good if the number of samples is large enough. We made a plot of error rate (using the rank mean  $\leq 10$ ) vs number of samples (not shown) and indeed the error rate decreases steadily.
- Even sub-sampling is sensitive to the choice of  $C$  (all the algorithms were). However, it still does well even in the hard margin case, so we think it is the best algorithm to adopt.
- LOO algorithms are reasonably good, but not as good as sub-sampling.
- The flipping algorithms were not as good as expected apart from FLIP-DIST (which Asa suggested) and FLIP-VALID (which of course uses a validation set).
- FLIP-VALID is actually the best method overall but of course cheats (uses extra information). Perhaps using a validation set can be approximated by using sub-sampling, but this makes the algorithm rather slow. It could still be worth using, though. It was also strange that taking the validation error itself did very badly (mean  $\leq 10$  was approximately 6!) but using

a sigmoid made the method the best method. Perhaps this should be applied to the methods that use  $\xi$ ?

- GRAD- $C$  is very good which we believe is due to it finding good values of  $C$  for the classifier itself as part of the algorithm.
- It is surprising that all the algorithms give rather similar error rates as when you compare them to the BAYES-GOD they are a long way off the optimum. However, they are not so far from SVM-GOD  $C$  which suggests that if you stay within the SVM/learning framework to perform data-cleaning you cannot do a lot better than these algorithms.

ALGORITHM	mean	rank 1	rank <= 2	rank <= 3	mean <= 10	mean <= 5
SVM- $\alpha$	6.69	31	43	53	4.49	3.13
SVM- $\xi$ $C$	4.64	36	53	68	3.68	2.72
SUB-ERR 300 2 $C$	4.93	39	57	64	3.76	2.69
SUB-ERR 300	5.03	39	58	63	3.77	2.69
LOO- $\xi$	6.28	35	53	60	4.18	2.86
LOO- $\xi$ $C$	4.76	37	53	65	3.73	2.75
FLIP-DIST $C$	4.74	36	53	63	3.75	2.76
FLIP-VALID $C$	4.39	38	58	72	3.38	2.56
GRAD- $C$	4.89	38	49	63	3.86	2.81
SVM-GOD	8.75	27	39	50	4.81	3.24
SVM-GOD $C$	4.07	41	60	70	3.39	2.57
BAYES-GOD	1.46	84	88	92	1.46	1.42

Figure 1: Comparison of algorithms on toy data. More detailed results of other algorithms run are shown in Figure 2. In all the algorithms letters after the name of the algorithm imply the following:  $C$  is the value of soft margin chosen using a validation set.  $2C$  means two times this value (as sub-sampling involves a smaller training set). No  $C$  implies a hard margin algorithm. In the sub-sampling algorithms 300 after the name of the algorithm indicates the number of sub-samples used.

We also performed an experiment measuring the test error obtained when using a soft margin SVM with the mislabeled example in the training set which gave 14.4% error ( $C = 0.53$ , chosen on the same validation set as testing) over 100 runs. Without the mislabeled point (which one could remove if one was God) an SVM gave 12.9% error ( $C = 0.34$ ). Thus there is a significant improvement when one removes the outlier. This was not obvious to us before as we believed a soft margin SVM would produce close to optimal results. We thus tried to estimate God’s performance and tried to remove one mislabeled point using the SVM- $\xi$   $C$  algorithm giving an error rate of 14.1% ( $C = 0.19$ ). Thus performance improved, but was unfortunately the gain was not as high as is possible given God’s performance: it is a pity such improvements or close to such

improvements cannot be found. We are only human beings, we won't ever be able to do as well as God )-: However, we suggest an automatic cleaning method where we assume that only one example is mislabeled and we systematically remove the top ranked example and retrain. Since we have a substantial number of cases (roughly two thirds) in which the top example is not mislabeled, maybe systematically removing it is not the best solution. However, finding a good automatic cleaning prescription is important, particularly because in reality we don't know how many if at all are mislabeled. We do not deal with these issues here, our report concentrates instead on the identification problem rather than the automatic cleaning one.

## 5.2 Real data

We then performed experiments on a colon cancer problem [1] and a leukemia problem [4] (see also [6] for a treatment of these problems). In the colon cancer problem 62 tissue samples probed by oligonucleotide arrays contain 22 normal and 40 colon cancer tissues that must be discriminated based upon the expression of 2000 features. In the leukemia problem 72 bone marrow and blood samples contain two variants of leukemia (47 ALL and 25 AML) that must be discriminated based upon the expression of 7129 features. (Note that in real data such as this although we deliberately mislabel points in the dataset it is possible points are already mislabeled.)

The results are shown in Figure 3 and Figure 4.

The results generally support the findings from the toy experiments. On the colon cancer dataset sub-sampling error rate seems again to be the best method. Note that we did not have enough data to make a validation set here so where algorithms required a soft margin (e.g SVM- $\xi$ ) we chose the median  $\alpha$  value as the value of  $C$ . Overall, the sub-sampling methods still proved to be the best methods to use, particularly in view as these are hard margin algorithms were you do not need to choose  $C$ . The *LOO* algorithms did not perform so well, nor did the flipping algorithms (worst results removed for clarity). GRAD-C did not perform as well as in the toy data but we believe this is because this problem does not depend so much on the value of soft margin.

On the leukemia data (Figure 4) the problem is much easier with most of the mislabeled points being identified and given a rank of 1. This justifies the use of data cleaning methods in micro-array experiments: many of the datasets are of the same level of complexity/difficulty as this dataset (actually in our experience the colon data seems to be a more unusual case). We noticed however that example number 66 in this dataset (from the version of the dataset posted on the web-site) is consistently mislabeled. During sub-sampling it is it is classified as the opposite label to the one given in the original dataset on every single sample. This leads to the conclusion that this data point is mislabeled, either in the original dataset or by accident since then. We have yet to follow this up. If this point is mislabeled then we note that we note that the SUB-ERR algorithm would correctly identify every possible single deliberate mislabeling were this corrected.

ALGORITHM	mean	rank 1	rank <= 2	rank <= 3	mean <= 10	mean <= 5
SVM- $\alpha$	6.69	31	43	53	4.49	3.13
SVM- $\xi$ $C^*$	6.57	29	47	56	4.31	3.04
<b>SVM-<math>\xi</math> C</b>	<b>4.64</b>	<b>36</b>	<b>53</b>	<b>68</b>	<b>3.68</b>	<b>2.72</b>
SUB-ERR 50 C	7.73	1	12	40	5.29	4.17
SUB- $\xi$ 50 C	5.35	39	53	65	3.71	2.74
SUB- $\alpha$ 50 C	4.89	37	53	65	3.74	2.77
SUB-ERR 300 2C	4.93	39	57	64	3.76	2.69
<b>SUB-<math>\xi</math> 300 2C</b>	<b>4.74</b>	<b>40</b>	<b>57</b>	<b>65</b>	<b>3.68</b>	<b>2.67</b>
SUB- $\alpha$ 300 2C	5.41	28	44	54	4.22	3.10
SUB-ERR 50	8.68	1	5	18	6.01	4.41
SUB- $\xi$ 50	6.24	32	51	61	4.13	2.91
SUB- $\alpha$ 50	6.43	30	45	53	4.48	3.13
<b>SUB-ERR 300</b>	<b>5.03</b>	<b>39</b>	<b>58</b>	<b>63</b>	<b>3.77</b>	<b>2.69</b>
SUB- $\xi$ 300	5.07	42	55	65	3.80	2.68
SUB- $\alpha$ 300	5.60	31	43	55	4.21	3.08
LOO- $\xi$	6.28	35	53	60	4.18	2.86
LOO- $W^2$	6.42	32	48	56	4.32	3.02
<b>LOO-<math>\xi</math> C</b>	<b>4.76</b>	<b>37</b>	<b>53</b>	<b>65</b>	<b>3.73</b>	<b>2.75</b>
LOO- $W^2$ C	20.62	8	11	12	7.90	4.52
FLIP- $W^2$	6.46	32	47	56	4.33	3.03
FLIP- $W^2$ C	13.08	16	25	29	6.70	3.98
FLIP-SPAN C	9.51	14	27	38	5.78	3.75
<b>FLIP-DIST C</b>	<b>4.74</b>	<b>36</b>	<b>53</b>	<b>63</b>	<b>3.75</b>	<b>2.76</b>
<b>FLIP-VALID C</b>	<b>4.39</b>	<b>38</b>	<b>58</b>	<b>72</b>	<b>3.38</b>	<b>2.56</b>
<b>GRAD-C</b>	<b>4.89</b>	<b>38</b>	<b>49</b>	<b>63</b>	<b>3.86</b>	<b>2.81</b>
SVM-GOD	8.75	27	39	50	4.81	3.24
SVM-GOD C	4.07	41	60	70	3.39	2.57
BAYES-GOD	1.46	84	88	92	1.46	1.42

Figure 2: Comparison of algorithms on toy data. Given in bold are some of the strongly performing algorithms. For clarity only these algorithms and the GOD algorithms are given in a separate table in Figure 1. In all the algorithms letters after the name of the algorithm imply the following:  $C^* = \frac{1}{2} \max_i(\alpha_i)$ ,  $C$  is the value of soft margin chosen using a validation set,  $2C$  is  $2 * C$ . In the sub-sampling algorithms 50 and 300 after the name of the algorithm indicate the number of sub-samples used. The maximum rank score is 100.

ALGORITHM	mean	rank 1	rank <= 2	rank <= 3	mean <= 10	mean <= 5
SVM- $\alpha$	8.48	15	19	25	4.77	3.54
SVM- $\xi$ $C^*$	7.45	13	14	20	4.82	3.87
SUB-ERR 300	7.14	16	19	27	4.59	3.51
SUB- $\xi$ 300	7.30	12	17	23	4.91	3.64
SUB- $\alpha$ 300	7.53	14	21	27	4.43	3.41
LOO- $\xi$	8.43	10	16	21	5.43	3.75
LOO- $\alpha$	7.43	7	13	18	5.03	3.96
FLIP-DIST $C^*$	7.17	10	12	17	4.93	3.96
GRAD-C	8.43	9	17	21	4.98	3.82

Figure 3: Comparison of algorithms on the colon cancer dataset. In all the algorithms letters after the name of the algorithm imply the following:  $C^*$  uses  $C$ = median  $\alpha$  value, In the sub-sampling algorithms 300 after the name of the algorithm indicate the number of sub-samples used. The maximum rank score is 62.

ALGORITHM	mean	rank 1	rank <= 2	rank <= 3	mean <= 10	mean <= 5
SVM- $\alpha$	1.66	29	71	71	1.66	1.63
SUB-ERR	1.98	65	71	71	1.20	1.13
SUB- $\alpha$	1.68	28	71	71	1.68	1.65
LOO-ERR	1.41	56	71	71	1.33	1.26
LOO- $W^2$	1.54	42	71	71	1.52	1.45

Figure 4: Comparison of algorithms on the leukemia dataset. In the sub-sampling algorithms 200 after the name of the algorithm indicate the number of sub-samples used. The maximum rank score is 72.

## 6 Conclusions

We introduced some very simple minded methods of performing data cleaning which appear to perform well enough to be used for other micro-array problems. Our final conclusion is that sub-sampling (in particular, SUB-ERR) is a good method to use and it also has the advantage of a naturally interpretable output which is useful for analysis of data anyway. We therefore decided to include it in the micro-array code library as the method of choice. We believe from the experiments with the GOD algorithms that it is not possible to improve these results a lot if one stays within the framework of SVMs, and perhaps this extends to the domain of supervised learning.

We suggest to use SUB-ERR (counting the average error of each training point over many sub-samplings) with whichever learning algorithm gives the best generalization performance. If the data is very simple (leukemia data)

such that correlation coefficients can do very well (see previous reports) then we suggest to use a simple algorithm such as LDA or other *average* case methods. Perhaps one should use even univariate methods. For more complex data where correlation coefficients are significantly worse than other methods (see previous reports) then we suggest using SVMs. If feature selection analysis indicates that feature selection improves generalization performance then we expect the data cleaning should be done by using a feature selection method as well.

We also think it is interesting to look at individual examples error rates to see if the error distribution is smooth, or if some points are consistently incorrectly classified by the algorithm (even in the case where they have the correct label). We believe that this is interesting just in terms of understanding the problem one is trying to solve, moreover if the examples are patients this gives you some extra information that classifiers and feature selectors do not give you concerning the individuals. SUB-ERR also gives exactly this kind of information.

## Acknowledgements

We thank Asa Ben-Hur for discussions. Asa suggested the algorithm **FLIP-DIST** in the family of “flipping” algorithms.

## References

- [1] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745–6750, 1999.
- [2] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing kernel parameters for support vector machines. *Machine Learning*, 2000. Submitted.
- [3] T. Furey, N. Cristianini, N. Duffy, M. Schummer, D. B. Lin, and D. Haussler. Support vector machine classification and validation of cancer tissue using microarray expression data.
- [4] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [5] I. Guyon, N. Matić, and V. N. Vapnik. Discovering informative patterns and data cleaning. In *AAAI workshop on Knowledge Discovery in Databases, KDD'94*, pages 145–156, Seattle, WA, July 1994. MIT Press.
- [6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 2000.
- [7] N. Matić, I. Guyon, and V. N. Vapnik. Computer aided cleaning of large databases for character recognition. In *AT&T unpublished report*, 1991.