Probabilistic Noise Identification and Data Cleaning

Jeremy Kubica and Andrew Moore

CMU-RI-TR-02-26

October 2002

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Real world data is never as perfect as we would like it to be and can often suffer from corruptions that may impact interpretations of the data, models created from the data, and decisions made based on the data. One approach to this problem is outlier detection or anomaly detection in which an algorithm identifies and removes entire suspect records. But if only certain fields in a record have been corrupted then useful uncorrupted data will also be thrown out. In this paper we present an approach for identifying corrupted fields and using the remaining non-corrupted fields for subsequent modeling and analysis. Our approach learns a probabilistic model from the data that contains three components: a generative model of the clean data points, a generative model of the noise values, and a probabilistic model of the corruption process. We provide an algorithm for the unsupervised discovery of such models and empirically evaluate both its performance at detecting corrupted fields and the resulting improvement this gives to a classifier.

Contents

1 Introduction

Real world data is never as perfect as we would like it to be and can often suffer from corruptions that may impact interpretations of the data, models created from the data, and decisions made based on the data. In this paper we present an approach for identifying corrupted fields and using the remaining non-corrupted fields for subsequent modeling and analysis. We consider the case where corrupted values are completely replaced by noise values during corruption. These errors may occur due to such factors as partial sensor failure, environmental conditions, transcription error, and data storage corruption. By identifying noisy values it may be possible to increase classification accuracy, improve models, or make better decisions.

We present an approach that uses the data to learn a probabilistic model containing three components: a generative model of the clean data points, a generative model of the noise values, and a probabilistic model of the corruption process. Thus we explicitly model both the noise and the data corruption process. This explicit modeling has two possible advantages. First, explicitly modeling the noise and the data corruption process may increase the accuracy and robustness of the noise identification. For example, attribute corruptions may not be independent. Noise in a camera image may affect multiple pieces of information being extracted from the image. Second, a model of the corruption process may be used to improve the data collection process. For example, in the case of robotic planning an explicit model of sensor failure can show that one or more sensors have an abnormally high noise rate. This information could then be used to plan better experiments or to choose the best (and presumably reduced) weights given to those sensors.

The input is assumed to consist of N data points each of which has D attributes. These attributes may be real valued, categorical, or a combination of both. Of the $N * D$ values, a certain fraction have been corrupted. Table 1 shows part of a typical dataset. The entries in italics represent values that have been replaced by some corruption process. It is important to notice that we are interested in cases where corruptions occur on the level of the individual values and not entire data points. We wish to label only the corrupted portions of the data points as such and to use all of the data to obtain better estimates of both the underlying model and the corruption process itself.

Att1	Att2	...
1.00	3.14	...
4.50	*2.00*	...
5.01	1.22	...
...	...	...

Table 1: Example of a corrupted dataset.

As motivation consider the points shown in Figure 1. The point marked with an x has had a single attribute corrupted. Despite this, the remaining uncorrupted attribute can still be used to estimate the x-coordinate of the mean of the Gaussian. Further, the point marked with a triangle is a second outlier corrupted the same way, but could

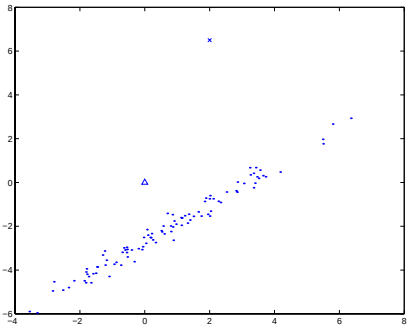

Figure 1: Example of two points each with a single corrupted attribute.

have plausibly resulted from corruptions in either or both dimensions. By modeling the corruption process we may be able to use knowledge of the first point's corruption to more accurately determine which corruptions are present in the second point. Of course both of these advantages are more significant as the number of corrupted points and the number of dimensions increase.

2 Model Learning and Noise Identificiation

The ultimate task is to both identify the corrupted values and learn accurate models of the underlying generative process. Further, the information gained from these tasks can then be used to calculate "clean" values for the corrupted data. We approach this task by defining a probabilistic generative model for the data points and presenting an iterative approach for maximizing the model's log-likelihood.

2.1 Generative Model

We assume that points are generated independently by the model shown in Figure 2. Thus we currently do not consider the case of time dependent data or noise. Points are generated using three distinct models: the clean model, the noise model, and the corruption model. Point generation can thus be viewed as a two-stage process. In the generation stage, noise free data points are generated according to an underlying probabilistic model. The data is then corrupted. Which attributes are corrupted is determined by an underlying corruption probability model and the new values for the corrupted points are generated according to an underlying noise model.

We define the elements of our model as:

- *Clean Model* (M_C): The generative model for the uncorrupted data points.
- *Noise Model* (M_N): The generative model for the noise values.
- *Corruption Model* (M_R): The probabilistic model for which values are corrupted by noise.

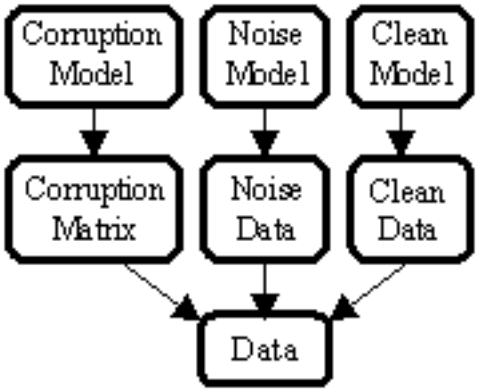

Figure 2: Probabilistic model of data generation.

We call the combination of the corruption model and the noise model our *noise system* and the combination of all three models the *generative model*. Further, we refer to the Boolean matrix marking whether or not a value has been corrupted as the *corruption matrix* (CM) and the correspond Boolean vector for a data point a *corruption vector*.

It is important to appreciate that in the model above, we do not restrict ourselves to specific classes of probability models. This is done intentionally to provide flexibility in the model and allow the algorithm to work on a wide range of underlying models. Further, note that the use of an arbitrary corruption model allows us the possibility to account for dependent corruptions.

2.2 Iterative Identification of Noisy Values

We want to learn the underlying models for which the data and learned corruption matrix are most likely. Thus, the problem reduces to:

$$\underset{M_C, M_N, M_R, CM}{argmax} P(Data,\, CM | M_C, M_N, M_R) \tag{1}$$

where we note:

$$\begin{aligned} P(Data,\, CM | M_C, M_N, M_R) = \\ P(Data | M_C, M_N, CM) P(CM | M_R) \end{aligned} \tag{2}$$

because CM is conditionally independent of M_C and M_N given M_R. This optimization can be interpreted two ways. First, the corruption matrix itself is a piece of hidden data and thus we are trying to find the models for which all of the data is most likely. Second, the corruption matrix is part of the set of models we are trying to learn, but we are constraining it to be probable with respect to the corruption model. Thus, the hope is the corruption matrix will not be arbitrarily complex, but rather will contain some underlying structure.

We could in theory use an exhaustive search algorithm that tries each possible corruption matrix. At each setting, the appropriate underlying models could be learned

using standard MLE techniques and the log-likelihood of the data could then be calculated. In the end the corruption matrix and corresponding models that produced the best log-likelihood would be selected. Unfortunately this approach requires searching over $O(2^{N*D})$ corruption matrices and is computationally infeasible for all but the smallest data sets.

Instead we propose an iterative approach that consists of alternating between learning the underlying generative models and learning the corruption matrix. We turn the optimization into a hill-climbing problem over the domain of corruption matrices. Using the model given above, we can break our task into two steps:

1. For a fixed corruption matrix, we can learn the probabilistic models.

2. For fixed generative models, we can estimate the corruption matrix.

Step 1 reduces the problem of learning models from data with corrupted values to the problem of learning models from data with missing values. The clean model can be learned by treating marked fields in the corruption matrix as missing. Depending on the classes of the models used, this can effectively be done using an EM based approach such as the one described in [Ghahramani and Jordan, 1994] for Gaussian mixture models. Similarly the noise model can also be learned from the data points, this time using the inverse of the corruption matrix to mark which fields are missing. Finally, learning the corruption model simply consists of learning a probability function over Boolean vectors.

Step 2 consists of finding a corruption matrix or "surprising" values given fixed probabilistic models. Since the points are assumed to be independent, we can find the corrupted fields for one point at a time. Below we discuss techniques to estimate the corruption vector of a point.

Thus, iterating over these two steps gives us the following algorithm:

1. Start assuming no corrupted points.

2. Until some termination criterion is met:

   (a) Learn the underlying models using the current corruption matrix.

   (b) Update the corruption matrix using the current models.

3. Return the final corruption matrix.

2.3 Identification of Noisy Values

There are a variety of methods for estimating a point's corruption vector given the underlying generative models. We present several such search methods below. We use R to refer to the corruption vector for a given point x under consideration and R_d to refer to the value of the dth attribute of this vector. We use M to refer to all of the models (M_C, M_N, and M_R). Thus $P(x, R|M)$ is the probability of a point and its corruption vector given the generative models.

2.3.1 Full Search

From (1) we note that we would like to find:

$$\underset{R}{argmax}\ P(R,\, x, |M) = \underset{R}{argmax}\ P(R|x,\, M) \tag{3}$$

where the equality holds because $P(x|M)$ does not depend on R. Again, it is possible to use an exhaustive approach to solve (3) by iterating over each of R's 2^D possible settings. Unfortunately, this approach is not computationally feasible for high dimensional datasets. It is important to note that we can also approximately solve (3) using a hill climbing search over R.

2.3.2 Greedy Search

In order to combat the above problem of intractability, we can consider a greedy approach that limits the search to look at each attribute in turn. The probability that the dth attribute is corrupted (that is $R_d = 1$) given the point and models is:

$$P(R_d = 1|x,\, M) = \frac{\sum_{R\ s.t.\ R_d=1} P(x|R,\, M)P(R|M)}{\sum_R P(x|R,\, M)P(R|M)} \tag{4}$$

Again these sums are over 2^{D-1} elements and may not be computationally feasible. Fortunately, we can quickly approximate this probability by assuming that the previous corruption vector holds for the other $D-1$ attributes:

$$P(R_d = 1|x,\, M) = \frac{P(x, R_{(d=1)}, M)}{P(x, R_{(d=1)}, M) + P(x, R_{(d=0)}, M)} \tag{5}$$

where the symbol $R_{(d=v)}$ denotes the vector R with the dth attribute set to v and the other elements set to their previous values.

We can mark the dth attribute of the point corrupt if $P(R_d = 1|x,\, M)$ exceeds some threshold or simply if $P(R_d = 1|x,\, M) > P(R_d = 0|x,\, M)$. This approximation agrees with intuition. For each point we are asking: "Given that I believe that this corruption vector holds, what is the probability that this attribute is corrupt." More importantly, this approximation makes the search computationally tractable.

2.3.3 Randomized Greedy Search

It is also possible to produce a randomized version of the greedy algorithm. Specifically, we can mark each value as corrupt with probability $P(R_d = 1|x,\, M)$ as defined in (5). Including a small minimum probability of flipping a bit in the corruption vector it may also help the algorithm escape local minima. As shown below the randomized nature of this approach can aid the optimization and produce better results.

3 Noise Correction

The models above can be directly employed to correct the corrupted data points. Specifically, given a data point and corresponding corruption vector we can "correct" the data

point by determining the most likely values for the corrupted attributes given the uncorrupted attributes. This can be done using an EM approach.

More importantly, this approach can be applied to previously unseen data points. The above techniques can be used to estimate a point's corruption vector and correct the appropriate values. Thus we can build the models on a small portion of the data and use them to clean the remainder of the data or to check for corruptions online.

4 Related Work

There is a significant interest in noise identification and data cleaning in the field of computer science. One technique which has seen a significant amount of work is to consider noise on a pointwise scale and remove noisy data points [Guyon *et al.*, 1996], [Gamberger *et al.*, 1999], [Brodley and Friedl, 1996], [John, 1995], [Arning *et al.*, 1996] and others. For example, in [Arning *et al.*, 1996] Arning *et. al.* present a linear time method for detecting deviations in a database. They assume that all points should be similar, which may not be true in unsupervised learning tasks, and that an entire point is either noisy or clean. Assuming entire points are noisy or clean is also common in outlier and novelty detection [Huber, 1981] and [Hampel *et al.*, 1985]. A significant downside to looking at noise on a pointwise scale is that entire points are thrown out and useful uncorrupted data may be lost. In datasets where almost all points have at least a few corruptions this may prove disastrous.

Several domain independent models have been presented which examine the case of attributewise noise. Schwarm and Wolfman examined the use of Bayesian methods to clean data in [Schwarm and Wolfman, 2000]. They also use a probabilistic approach, but it differs from this paper in several respects: we do not assume a subset of precleaned instances and we use an iterative approach that considers other corruptions that may be present in a data point. Further, our model includes explicit models of both the noise and the corruption process allowing us to take advantages of dependencies in the corruption process and regularities in the noise. In [Teng, 1999] Teng presents a data cleaning method designed to improve accuracy on classification tasks. This model only looks at points which have been misclassified and attempts to make corrections based in part on the predicted attribute value given the rest of the point (which itself may be noisy). Again there is no attempt to model the noise or the corruption process. Finally, in [Maletic and Marcus, 2000] Maletic and Marcus describe a method for finding attributewise outliers. This method does not account for or utilize possible dependencies between the attributes.

Speech recognition and signal separation is another area that has seen a lot of work in data cleaning and noise modeling, including [Attias *et al.*, 2001], [Roweis, 2000], [Raj *et al.*, 2000] and others. While these methods have been shown effective, they often make limiting assumptions, such as time dependencies between points or pretrained models and classifiers. For example, [Roweis, 2000] Roweis presents a refiltering method designed to separate noise values from speech values in a signal, but requires the use of pretrained hidden Markov models for each speaker and data with time dependencies.

Finally, there has been a significant amount of recent work in data cleaning the field

of data warehousing and data/information quality management. [Raisinghani, 1999] and [Maletic and Marcus, 2000] and references there in present an overview of some of this work. Much of this work is concerned with solving problems that are different from ours, such as: identifying duplicate records, merging databases, and finding inconsistencies. Further, much of the work makes use of the fact that the databases have some known structure which can by utilized. For example, many of the errors in name fields will be in terms of spelling or formatting differences or errors.

5 Results

5.1 Naturally Corrupted Data Sets

The algorithm was tested on three real world datasets. These datasets contained "natural" corruptions that were not explicitly generated from the assumed models.

5.1.1 Leaf Data, Rock Data, and Results

The leaf and rock data consist of attributes extracted from a series of pictures of leaves and rocks respectively. All of the pictures were taken using a Nikon Coolpix 800 digital camera against a white background. A single lamp was used to provide primary lighting, but little was done to standardize lighting or prevent illumination from secondary lighting sources. One data point was then generated for each picture as the three number vector corresponding to the average of each pixel's RGB value. Two natural corruptions were introduced. First, some pictures were *red corrupted* by placing a red filter over the primary light source, which was expected to produce a corruption in the red attribute. Second, some pictures were *black corrupted* by taking a picture without removing the lens cap, which was expected to corrupt all three attributes. Both datasets were modeled using: a single 3-dimensional Gaussian as the clean model, uniform noise (with the same range as the observed data) as the noise model, and a naive Bayesian model as the corruption model.

The leaf dataset consists of attributes extracted from 71 pictures of two classes of leaves: living and dead. As expected, the living leaves were green in color while the dead leaves were brownish or yellow. Of the 35 pictures of living leaves, 4 were red corrupted and 1 was black corrupted. Of the 36 pictures of dead leaves, 4 were red corrupted.

The rock dataset consists of attributes extracted from 56 pictures of two classes of rocks: slate and granite. The granite rocks contained sufficient feldspar to give them a slightly orange coloring. Of the 28 pictures of granite, 4 were red corrupted. Of the 28 pictures of slate, 4 were red corrupted and 1 was black corrupted.

The algorithm was run on the leaf data set for 2,000 iterations using randomized greedy search and a minimum flip probability of 0.01. The resulting output found 10 corruptions including the 8 corruptions in each of the red values for the pictures taken under the red filter and corruptions in both the red and green values of the picture taken with the lens cap on. Thus the results agreed almost exactly with those predicted from knowledge of the experiment.

The algorithm was run on the rock data set for 5,000 iterations using randomized greedy search and a minimum flip probability of 0.05. The resulting output found 27 corruptions of which 22 were in data points known to be corrupted and 4 were in data points that were not intentionally corrupted. Of the 9 corrupted data points: 6 were found to be completely corrupted (all three attributes), 1 was found to be corrupted in only the red attribute, and 2 were found to be corrupted in the red and blue attributes.

Further, in order to test the noise identification's effect on classification, Leave-One-Out-Cross-Validation tests were performed on both sets of data using various assumptions. Each test consisted of fully relearning the models and corruption matrix from their default values. The results are shown in Table 2

Search Method	Leaf Error	Rock Error
Assume Noiseless	17	11
Randomized Greedy	7	8
Greedy (1 itr)	16	10
Greedy (10 itr)	16	10
Greedy (2000 itr)	16	10

Table 2: Results for different learning assumptions and optimization times.

From the results we see that accounting for corruptions, as with the randomized greedy algorithm, can lead to significant improvement in classification accuracy. Note that the model assuming noiseless points is the same as a Bayesian classifier that does not account for noise. It is important to note that at no time does the optimization consider the classification task itself. The algorithm simply tries to learn clean models (one for each class), a noise model, and a corruption model that give the best log-likelihood.

Further, the above results demonstrate one possible advantage of the randomized strategy. Namely, the randomized strategy has the potential to escape local minima and result in a better ultimate performance.

5.1.2 Webpage Data and Results

The webpage data consists of site counter data: the number of page views and time spent on site, for two personal websites. The "corruptions" arise naturally from abnormal viewing patterns, such as spending a large amount of time looking at few pages and spending a short amount of time looking at a many pages. The points are shown in Figure 3.

The algorithm was run on the webpage data for 1000 iterations using randomized greedy search and a minimum flip probability of 0.01. The clean model was assumed to be a mixture of 3 Gaussians, the noise model was assumed to be uniform noise (with the same range as the observed data), and the corruption model was assumed to be a naive Bayesian model. Since some values were repeated several times, a small amount of random noise [0.0,0.1] was added to all values to prevent the covariance matrices

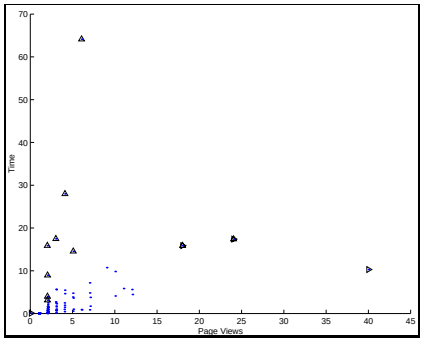

Figure 3: Web data with corruptions marked.

from becoming singular.

The results are also shown in Figure 3. Corrupted points are marked with one or two triangles that point along the axis in which the corruption was found. This result demonstrates the ability to identify noise on an attribute basis. For example, the point that includes the longest amount of time only looks at an average number of pages. Further, these results demonstrate an application of the algorithm to a completely unsupervised learning task.

5.2 Artificial Corruptions vs. Cleaning

The algorithms were also tested by tracking how many errors in artificially corrupted datasets could be identified. Two different test sets were used: a noise free version of the rock data described above and the UCI Iris Dataset [Blake and Merz, 1998]. Noise was generated by choosing to corrupt each point with some probability p. For each point chosen for corruption, a corruption vector and a noise vector were sampled from a joint Bayesian corruption model and a uniform noise model respectively. Parameters for the joint model were generated randomly at the start of each iteration.

The algorithm's success was measured using percent improvement:

$$Per.\,Improve = \frac{Errs_{Before} - Errs_{After}}{Errs_{Before}}$$

where the error in a corruption matrix is computed by saving the actual corruption matrix (CM_A) and comparing it with the learned corruption matrix (CM_L):

$$Errs = \sum_{i=1}^{N}\sum_{j=1}^{D} \mid CM_A(i,j) - CM_L(i,j) \mid$$

This measure of performance was used for two reasons. First, since the points were corrupted randomly, the actual number of corruptions on a given trial varied. Secondly, this method penalizes any false positives.

Figures 4 and 5 show the percent improvement versus p on artificially corrupted rock and iris data respectively. The solid line shows the 95% confidence interval for

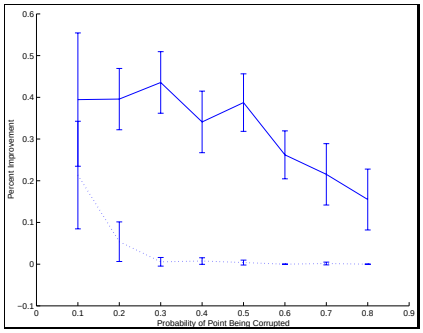

Figure 4: Percent improvement on artificially corrupted rock data.

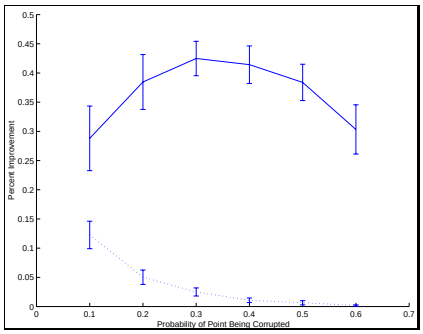

Figure 5: Percent improvement on artificially corrupted iris data.

the results of the algorithm using randomized greedy search. The rock and iris data used 1000 and 2000 optimization iterations respectively. The dashed line shows the 95% confidence interval for the results of the algorithm using greedy search and 1 optimization iteration.

Both figures show relatively strong performance of the randomized algorithm even for medium amounts of noise. It is important to note that since the noise values were sampled from a uniform distribution, many corrupted points may appear uncorrupted. As expected, performance begins to decline as the amount of noise increases. Further, the figures show a significant advantage to using a randomized iterative approach to a single pass for estimating which points are corrupt.

5.3 Model Accuracy

Finally, we examined the problem of learning an accurate model given a large amount of noise. Basically, we were interested in whether "cleaning" the data resulted in a more accurate estimate of the models parameters than would result from simply throwing out the corrupted points. Intuitively this should be the case since corrupted points may still contain uncorrupted attributes.

50 data points were artificially generated from a 4-dimensional Gaussian. The first

Estimation Method	Dist. From True Mean
True Mean	0.00
Only Noiseless Points	1.54
Cleaned Data (joint M_R)	0.71
Cleaned Data (naive M_R)	1.86

Table 3: Model parameter estimation from noisy dataset.

20 points were then corrupted in the 2nd and 3rd dimensions. The second 20 points were corrupted in the 1st and 4th dimensions. The final 10 points were left uncorrupted. All uncorrupted values were in the range [-10.0, 10.0] while the corrupted values were either in the range [-20.0, -15.0] or [15.0, 20.0].

The algorithm was run for 2000 iterations using randomized greedy search and a minimum flip probability of 0.04. The clean model was assumed to be a single Gaussian and the noise model was assumed to be uniform noise. Both joint and naive corruption models were tested in order to see whether the joint model would be able to use the dependent nature of the corruptions.

As an estimation of performance we examined the Euclidean distance of resulting estimated means with the true mean from the noiseless data. The results are shown in Table 3. As shown, using the 40 noisy points resulted in a better estimate of the mean than simply throwing them out and calculating the mean from the remainder. Further, as shown the joint corruption model was able to use the dependent nature of the corruptions to obtain better performance.

6 Conclusions

This paper presents an iterative and probabilistic approach to the problem of identifying, modeling, and cleaning data corruption. The strength of this approach is that it builds explicit models of the noise and the corruption process, which may be used to facilitate such tasks as data cleaning and planning the collection of future data points. We show that this approach can lead to benefits in classification, overall data accuracy, and model accuracy using both real world and artificial data.

7 Acknowledgements

Jeremy Kubica is supported by a generous grant from the Fannie and John Hertz Foundation.

References

[Arning *et al.*, 1996] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A linear method for deviation detection in large databases. In *Knowledge Discovery*

and Data Mining, pages 164–169, 1996.

[Attias *et al.*, 2001] Hagai Attias, Li Deng, Alex Acero, and John C. Platt. A new method for speech denoising and robust speech recognition using probabilistic models for clean speech and for noise. In *Proc. of the Eurospeech Conference*, 2001.

[Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[Brodley and Friedl, 1996] Carla E. Brodley and Mark A. Friedl. Identifying and eliminating mislabeled training instances. In *AAAI/IAAI, Vol. 1*, pages 799–805, 1996.

[Gamberger *et al.*, 1999] Dragan Gamberger, Nada Lavrač, and Ciril Grošelj. Experiments with noise filtering in a medical domain. In *Proc. 16th International Conf. on Machine Learning*, pages 143–151. Morgan Kaufmann, San Francisco, CA, 1999.

[Ghahramani and Jordan, 1994] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

[Guyon *et al.*, 1996] Isabelle Guyon, Nada Matic, and Vladimir Vapnik. Discovering informative patterns and data cleaning. In *Advances in Knowledge Discovery and Data Mining*, pages 181–203. 1996.

[Hampel *et al.*, 1985] F. R. Hampel, P. J. Rousseeuw, E. M. Ronchetti, and W. A. Stahel. *Robust Statistics: The Approach based on Influence Functions*. Wiley International, 1985.

[Huber, 1981] Peter J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

[John, 1995] George H. John. Robust decision trees: Removing outliers from databases. In *Knowledge Discovery and Data Mining*, pages 174–179, 1995.

[Maletic and Marcus, 2000] Jonathan I. Maletic and Aaron Marcus. Data cleansing: Beyond integrity analysis. In *Proceedings of the Conference on Information Quality*, pages 200–209, 2000.

[Raisinghani, 1999] Vijay T. Raisinghani. Ceaning methods in data warehousing. Seminar Report, 1999.

[Raj *et al.*, 2000] Bhiksha Raj, Michael L. Seltzer, and Richard M. Stern. Reconstruction of damaged spectrographic features for robust speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*, 2000.

[Roweis, 2000] Sam Roweis. One microphone source separation. In *Neural Information Processing Systems*, volume 13, pages 793–799, 2000.

[Schwarm and Wolfman, 2000] Sarah Schwarm and Steve Wolfman. Cleaning data with bayesian methods. 2000.

[Teng, 1999] Choh Man Teng. Correcting noisy data. In *Proc. 16th International Conf. on Machine Learning*, pages 239–248. Morgan Kaufmann, San Francisco, CA, 1999.