Probabilistic Noise Identification and Data Cleaning

Jeremy Kubica
Carnegie Mellon University
Robotics Institute
Pittsburgh, PA 15213
jkubica@ri.cmu.edu

Andrew Moore
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213
awm@cs.cmu.edu

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 to identify and remove records that contain corruptions. Unfortunately, if only certain fields in a record have been corrupted then usable, uncorrupted data will be lost. In this paper we present LENS, an approach for identifying corrupted fields and using the remaining non-corrupted fields for subsequent modeling and analysis. Our approach uses the data to learn a probabilistic model containing three components: a generative model of the clean records, 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, as one example application, the resulting improvement this gives to a classifier.

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 LENS (Learning Explicit Noise Systems), an approach for identifying corrupted fields and using the remaining non-corrupted fields for subsequent modeling and analysis. We consider the case where a corruption completely replaces the original data value. Such errors could be the result of: partial sensor failure, environmental conditions, transcription error, or 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 of the complete data generation process. We explicitly model both the noise and the data corruption process giving several 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 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 N records with D real, categorical, or mixed attributes. Formally, each *record* is an entry in the data and corresponds to a single observation, item, or data point. Each *attribute* corresponds to a single field in the record. A *cell* refers to a single attribute of a single record. Thus the entire data set contains $N * D$ cells, certain fraction of which have been corrupted by noise. It is important to appreciate that we are interested in cases where corruptions occur on the level of the individual cells and not entire records or attributes. We wish to label only the corrupted portions of the records as such and to use all of the data to obtain better estimates of both the underlying model and the corruption process itself.

Figure 1 provides a motivating example. Although the record marked with an X has a corrupted attribute, the remaining uncorrupted attribute can be used in estimating the x-coordinate of the mean. Further, the record marked with a triangle is a second outlier that has been corrupted the same way, but could have plausibly resulted from corruptions in either or both attributes. By modeling the corruption process we may be able to use knowledge of the first record's corruption to more accurately determine which corruptions are present in the second record. Both of these advantages are more significant as the number of corrupted records and attributes increase.

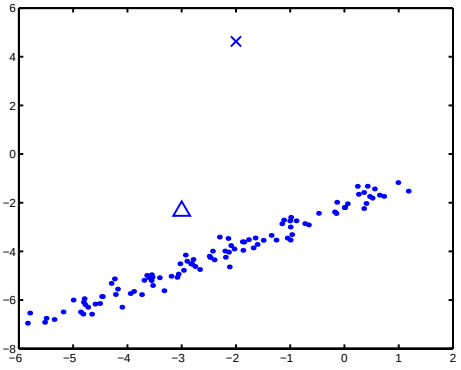

Figure 1. Example of two records each with a single corrupted attribute.

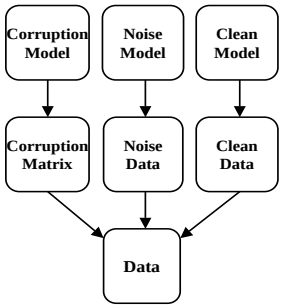

Figure 2. Model of data generation.

2 Model Learning and Noise Identificiation

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

2.1 Generative Model

We assume that records are generated independently by the model shown in Figure 2. Consequently we currently do not consider the case of time dependent data or noise. Records are generated using three distinct models: the clean model, the noise model, and the corruption model. Record generation can be viewed as a two-stage process. In the generation stage, (noise free) records are generated according to an underlying "clean" probabilistic model. The data is then corrupted. Which cells are corrupted is determined by an underlying corruption probability model and the new values for the corrupted cells 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 records.
- *Noise Model* (M_N): The generative model for the noise values.
- *Corruption Model* (M_R): The probabilistic model for which attributes are corrupted by noise.

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 cell has been corrupted as the *corruption matrix* (*CM*) and the correspond Boolean vector for a record a *corruption vector*.

It is important to appreciate that we do not restrict ourselves to specific classes of probability models. This is done intentionally to provide flexibility and allow the algorithm to work on a wide range of underlying models. Further, 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:

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

where:

$$P(Data, CM|M_C, M_N, M_R) = P(Data|M_C, M_N, CM)P(CM|M_R) \quad (2)$$

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 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.

Due to the large number, $O(2^{N*D})$, of corruption matrices it is computationally infeasible to exhaustively search for the best corruption matrix. 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. Using the model given above, we can break our task into two steps:

1. For a fixed corruption matrix, learn the models:

$$\underset{M_C,M_N,M_R}{argmax} P(Data, CM|M_C, M_N, M_R) \quad (3)$$

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

$$\underset{CM}{argmax} P(Data|M_C, M_N, CM)P(CM|M_R) \quad (4)$$

Step 1 reduces the problem to that of learning models from data with missing values. The clean model can be learned by treating marked cells in the corruption matrix as missing. Depending on the models used, this can effectively be done using an EM based approach such as the one described in [7] for Gaussian mixture models with missing data. Similarly the noise model can also be learned from the records, this time using the complement of the corruption matrix to mark which cells are missing. Finally, learning the corruption model simply consists of learning a probability distribution over Boolean vectors.

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

It should also be mentioned that it is possible to suggest an alternative approach that uses an EM method to learn the models while treating the corruption matrix as hidden data. Real valued entries could be stored in the corruption matrix to track the probability that a given cell is corrupted. Unfortunately, this approach quickly becomes computationally infeasible for many interesting models. For example, as an extreme case consider learning a full joint corruption model from a real valued corruption matrix. Simply accounting for a *single* record's corruption vector would take time $O(2^D)$. Further, this does not account for the cost of estimating the corruption matrix or learning the other two models.

2.3 Identification of Noisy Values

There are a variety of methods for estimating a record's corruption vector given the underlying generative models. From (1) we note that we would like to find:

$$\underset{R}{argmax} P(R, x|M) = \underset{R}{argmax} P(R|x, M) \quad (5)$$

where R is the corruption vector for a given record x under consideration and M is the collection all of the models (M_C, M_N, and M_R).

Again it is possible to use an exhaustive approach to solve (5) by iterating over each of R's 2^D possible settings. Unfortunately, this approach is not computationally feasible for high dimensional data sets. Instead, one possible solution is to approximately solve (5) using a hill climbing search over R.

We consider a greedy approach that limits the search to look at each attribute in turn, asking whether this attribute in this record is corrupt. Let R_d indicate the value of the dth attribute of this vector (where $R_d = 1$ indicates that the cell is corrupt and $R_d = 0$ indicates that it is not). The probability that the dth attribute is corrupted (that $R_d = 1$) given the record and models is:

$$P(R_d = 1|x, M) = \frac{\sum_{R s.t. R_d=1} P(x|R, M)P(R|M)}{\sum_R P(x|R, M)P(R|M)} \quad (6)$$

Again these sums are over at least 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:

$$P(R_d = 1|x, M) = \frac{P(x, R_{(d=1)}|M)}{P(x, R_{(d=1)}|M) + P(x, R_{(d=0)}|M)} \quad (7)$$

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 record corrupt if $P(R_d = 1|x, M)$ exceeds some threshold or simply:

$$R_d = \begin{cases} 1 & if\ P(R_d = 1|x, M) \geq P(R_d = 0|x, M) \\ 0 & if\ P(R_d = 1|x, M) < P(R_d = 0|x, M) \end{cases} \quad (8)$$

For each record we are effectively asking: "Given that I believe that this corruption vector holds, what is the probability that this attribute is corrupt?" Thus we are doing a hill-climbing search that does not blindly try neighbors, but rather uses information about the current corruption vector to direct the step to a neighboring corruption vector.

It is also possible to randomize the search to possibly aid optimization and escape local minima. Specifically, we can mark each cell as corrupt with probability $P(R_d = 1|x, M)$ as defined in (7). Including a small minimum probability of flipping a bit in the corruption vector may also help the algorithm escape local minima.

3 Noise Correction

The models above can be directly employed to correct the corrupted records. Specifically, given a record and corresponding corruption vector we can "correct" the record by using EM to determine the most likely values for the corrupted cells given the uncorrupted cells.

More importantly, this approach can be applied to previously unseen records. The above techniques can be used to estimate a record's corruption vector and correct the appropriate cells. 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 that has seen a significant amount of work is to consider noise on the record/point scale and remove noisy records such as presented in [1, 4, 8, 6, 11] and others. For example, in Arning *et. al.* present a linear time method for detecting deviations in a database [1]. They assume that all records should be similar, which may not be true in unsupervised learning tasks, and that an entire record is either noisy or clean. Assuming entire records are noisy or clean is also common in outlier and novelty detection [9, 10]. A significant downside to looking at noise on the scale of records is that entire records are thrown out and useful, uncorrupted data may be lost. In data sets where almost all records have at least a few corrupted cells this may prove disastrous.

Several domain independent models have been presented which examine the case of noisy cells. Schwarm and Wolfman examined the use of Bayesian methods to clean data [16]. 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 record. 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. Teng presents a data cleaning method designed to improve accuracy on classification tasks [17]. This model only looks at records which have been misclassified and attempts to make corrections based in part on the predicted attribute value given the rest of the record (which itself may be noisy). Again there is no attempt to model the noise or the corruption process. Finally, Maletic and Marcus survey several methods for finding outlier cells including statistical methods, association rules, and pattern-based methods [12]. The methods they describe do not account for or utilize possible dependencies between the attributes and do not build a model of the corruption process.

Speech recognition and signal separation are other areas that have seen work in data cleaning and noise modeling, including [2, 14, 15] 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 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 [15] .

There has been a significant amount of recent work in data cleaning the field of data warehousing and data/information quality management. [12, 13] 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 attribute will be in terms of spelling or formatting errors.

Finally, the AutoClass system is also similar to LENS in that it performs unsupervised clustering while allowing different classes to use different models for their attributes [5]. Determining which models to use for a given class is done in a step similar to the one described in section 2.2. In contrast to LENS, AutoClass appears to generate all records from a given class from the models for that class. As a result it does not appear to consider noise on the level of cells, where a single cell may be generated by a different model than cells of the corresponding attribute for the other records from this class.

5 Evaluation

5.1 Naturally Corrupted Data Sets

As an initial test, we ran LENS on several real world data sets. These data sets contained "natural" corruptions that were not explicitly generated from the assumed models.

5.1.1 Leaf and Rock Data

The leaf and rock data, summarized in Table 1, consist of attributes extracted from a series of pictures of leaves and rocks respectively. The leaf data set contains 71 records from pictures of living and dead leaves. As expected, the living leaves were green in color while the dead leaves were brownish or yellow. The rock data set contains 56 records from pictures of slate and granite. The granite rocks contained sufficient feldspar to give them a slightly orange coloring. All pictures were taken against a white background with a single lamp providing primary lighting, but little was done to standardize lighting conditions. One record 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: some pictures were *red corrupted* by placing a red filter over the primary light source and some pictures were *black corrupted* by taking a picture without removing the lens cap (thereby producing a picture of black pixels).

Table 1. Leaf and Rock data sets.

	TOTAL	RED	BLACK
LIVING	35	4	1
DEAD	36	4	0
GRANITE	28	4	0
SLATE	28	4	1

5.1.2 Leaf/Rock Cleaning Results

The settings for both data sets were chosen heuristically. Both data sets were initially 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 joint Bayesian model as the corruption model.

The LENS algorithm was run on the leaf data set for 2,000 iterations using randomized greedy search and a minimum flip probability of 0.02. The resulting output found 12 corruptions including 7 of the 8 corruptions in the R values for the red-corrupted pictures, corruptions in the B and G values of the 8th red corrupted pictures, and corruptions in all three values of the black corrupted picture. Thus the results agreed almost exactly with those predicted from knowledge of the corruptions.

The LENS algorithm was run on the rock data set for 500 iterations using randomized exhaustive search, a joint corruption model, and a minimum flip probability of 0.01. The algorithm found 27 corruptions, marking all three values of the 9 corrupted records as corrupt. Since only entire records were labeled as corrupt, later experiments used a naive corruption model and a randomized greedy search with 5,000 iterations and a minimum flip probability of 0.05. These settings only identified one corrupt record, the picture taken with the lens cap on, but showed promise in aiding the classification task discussed below.

It should be appreciated that the corruptions in the above data are very simple and could be identified by hand. This allows us to compare the resulting corruptions found with a known ground truth. Further, since both the data and the corruptions are generated by a real world process, we do not force either to be generated from the assumed models.

5.1.3 Leaf/Rock Classification Results

Since the data sets above contain class information, we are able to test whether noise identification leads to an improvement on classification. Leave One Out Cross Validation tests were performed on both sets of data. Each test consisted of: removing a record from the training set, fully relearning the models and corruption matrix from their default values, and classifying the removed record. The results are shown in Table 2. In addition to the randomized LENS algorithm, we also tested the following algorithms:

Table 2. Number of LOOCV errors for different noise identification methods.

SEARCH METHOD	LEAF ERROR	ROCK ERROR
ASSUME NOISELESS	17	11
RANDOMIZED GREEDY	6	6
GREEDY (1 ITR)	17	10
GREEDY (10 ITR)	17	10
GREEDY (500 ITR)	17	10
SIMPLE CELL	11	14
SIMPLE RECORD	6	10

- *Assume Noiseless* - assuming all records were noiseless:
$$R_d = 0 \ \ \forall d \tag{9}$$
- *Greedy* - a pure greedy variation (no randomization) of the LENS algorithm run for 1, 10, and 500 iterations as given in (7) and (8);
- *Simple Cell* - a non-iterative approach that determines whether a cell was more likely to have been generated from the learned clean model or a uniform noise model:
$$R_d = \begin{cases} 1 & P(x_d|x, M_N) \geq P(x_d|x, M_C) \\ 0 & P(x_d|x, M_N) < P(x_d|x, M_C) \end{cases} \tag{10}$$
This approach does not relearn the models given the corruption matrix; and
- *Simple Record* - a non-iterative approach that determines whether an entire record was more likely to have been generated from the learned clean model or a uniform noise model:
$$R = \begin{cases} (1, 1, \ldots) & P(x|M_N) \geq P(x|M_C) \\ (0, 0, \ldots) & P(x|M_N) < P(x|M_C) \end{cases} \tag{11}$$
This approach does not relearn the models given the corruption matrix.

Assume Noiseless corresponds to the performance of the system if no noise identification had taken place. *Simple Cell* and *Simple Record* do not learn corruption or noise models. Thus *Simple Cell* is similar to the approach presented in [16] for a single iteration probabilistic noise identification and *Simple Record* is similar to some approaches used for full point noise identification.

The results in Table 2 indicate that accounting for corruptions and learning a model of the corruption process, as with the randomized greedy algorithm, can lead to an improvement in classification accuracy. Further, the randomized nature of the algorithm can help escape local minima

and result in a better ultimate performance. It is important to appreciate 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.

5.2 Artificial Corruption

We also compared the algorithms by their ability to identify artificial corruptions. Three different test sets were used: a noise free version of the rock data described above, the UCI Iris data set, and the UCI Wine data set [3]. Noise was generated by choosing to corrupt each record with some probability p. For each record chosen, corruption and noise vectors were sampled from their respective models. The corruption model was a joint Bayesian model for the rock and iris data and naive Bayesian model for the wine data. Parameters for the corruption models were generated randomly at the start of each iteration.

The algorithm's success was measured using percent improvement:

$$Improvement = \frac{Errs_{\text{Before}} - Errs_{\text{After}}}{Errs_{\text{Before}}} \qquad (12)$$

where the error in a corruption matrix is the Hamming distance between the learned corruption matrix (CM_L) and the actual one (CM_A):

$$Errs = \sum_{i=1}^{N} \sum_{j=1}^{D} | CM_A(i,j) - CM_L(i,j) | \qquad (13)$$

This measure of performance was used for two reasons. First, since the records were corrupted randomly, the actual number of corruptions on a given trial varied. Secondly, this method penalizes any false positives. An improvement of 0 corresponds to the results if no noise detection was done (assume noiseless).

Figures 3, 4, and 5 show the improvement versus p on artificially corrupted rock, iris, and wine data respectively. The solid line shows the 95% confidence interval for the results of LENS using randomized greedy search with 2500, 2500, and 5000 optimization iterations for the rock, iris, and wine data respectively. The dashed line shows the 95% confidence interval for the results of the Simple Record algorithm describe above. As mentioned above this approach is similar to approaches used for full record noise identification. Finally, the dotted line shows the 95% confidence interval for the results of the Simple Cell algorithm describe above. Again as mentioned above this approach is similar to the approach presented in [16].

On the rock and iris data sets, the LENS algorithm consistently performs significantly better than either of the other algorithms on small to medium amounts of noise. As expected, performance begins to decline as the amount of noise increases. In the tests using the wine data set the performance is superior to that of Simple Record, but on the later cases is not significantly better than that of Simple Cell. Here it is important to notice that since an axis aligned Gaussian and naive corruption model are being used, LENS is unable to take advantage of possible dependencies between the attributes. Despite this, these results demonstrate an advantage to learning a noise and corruption model and using them in the identification of noise. The figures also show a significant advantage to looking for corruptions on the scale of cells rather than records.

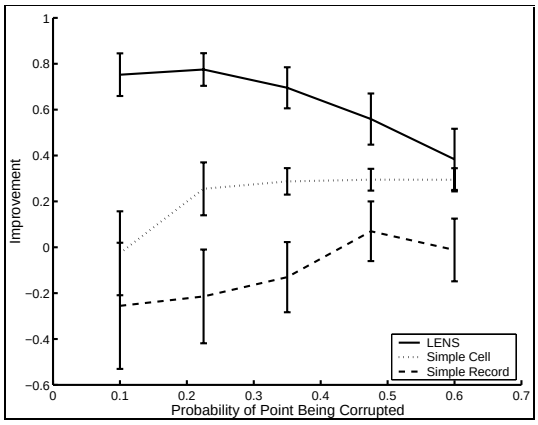

Figure 3. Percent improvement on artificially corrupted rock data.

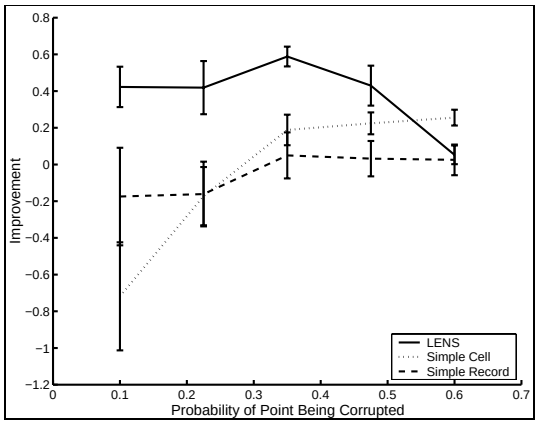

Figure 4. Percent improvement on artificially corrupted iris data.

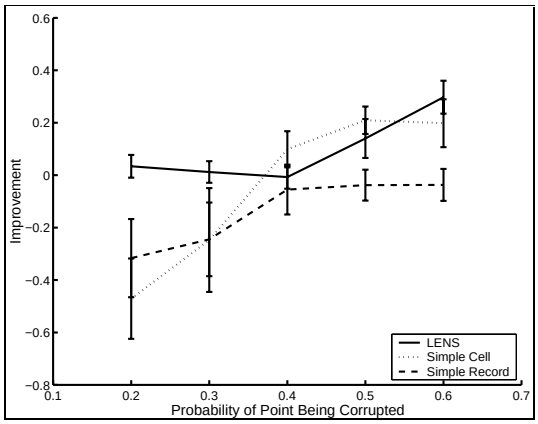

Figure 5. Percent improvement on artificially corrupted wine data.

Table 3. Error in parameter estimation from noisy data using various algorithms.

Estimation Method	Error DS1	DS2
All Records	3.20	1.95
Cleaned (joint M_R)	0.71	0.65
Cleaned (naive M_R)	1.86	2.14
Simple Cell	3.25	0.82
Simple Record	3.20	3.69
Noiseless Records	1.54	n/a

5.3 Model Accuracy

A final and important test is whether the learned clean model is accurate. In other words, we would like "cleaning" the data to result in a more accurate estimate of the model's parameters. To test this we examined corrupted data from two known clean distributions. In the first set (DS1) 50 records were artificially generated from a 4-dimensional Gaussian with the first 20 records corrupted in the 2nd and 3rd dimensions and the second 20 records corrupted in the 1st and 4th dimensions. In the second set (DS2) 80 records were artificially generated from a 4-dimensional Gaussian with each record corrupted (non-symmetrically) in exactly one attribute. Note that this means that 100% of the records were corrupted.

The LENS algorithm was then used to learn the parameters of the clean models. A Gaussian clean model, uniform noise model, and joint corruption model were assumed. For comparison, we also examined the performance in estimating the means after several other cleaning methods, including: no cleaning (assume noiseless), perfect record cleaning (all records containing any noise are removed), Simple Record from Section 5.1.3, Simple Cell from Section 5.1.3, and LENS with a naive corruption model. The estimation that removed all of the records containing at least one corruption did not use uncorrupted cells from corrupted records and thus produced results similar to those that would be obtained after perfectly filtering all outlier records.

Error was measured by the Euclidean distance of the estimated mean to the true mean that generated the data. The results are shown in Table 3. As shown, taking advantage of both dependencies in the corruptions and the uncorrupted cells in the corrupted records can result in a better performance at estimating the true underlying model. Also of interest is the performance of Simple Cell, which performed very well when records had at most a single corruption (DS2) and poorly when records contained multiple corruptions (DS1).

5.4 Comments on Evaluation

The above tests reflect an inherent difficulty in algorithm evaluation. Specifically, in order to evaluate a data-cleaning algorithm either the actual corruptions must be known or less direct evaluation, such as classification error, must be used. For example, all of the above tests used either simple and readily identifiable corruptions or synthetic corruptions generated from a known model. This was done in order to compare the discovered corruptions with a known ground truth.

6 Data Cleaning as Anomaly Detection

The LENS algorithm can also be run on uncorrupted data. In this case, abnormal points (or even points that simply do not conform to the clean model) may be identified as corrupted. In this way the LENS algorithm serves as an anomaly detector. Again, since we are identifying corruptions at the level of attributes, we can mark specific attributes as anomalous and give an "explanation" as to why a point may contain anomalies.

The webpage data consists of site counter data: the number of page views and time spent, for two personal websites. Here the "corruptions" correspond to abnormal viewing patterns, such as spending a large amount of time looking at few pages. The records are shown in Figure 6. LENS was run on the data for 1000 iterations using randomized greedy search and a minimum flip probability of 0.01. The clean, noise and corruption models were assumed to be a mixture of 3 Gaussians, uniform noise, and a naive Bayesian model respectively. Since some values were repeated several times, a small amount of random noise [0.0,0.1] was added to all values to prevent singular covariance matrices.

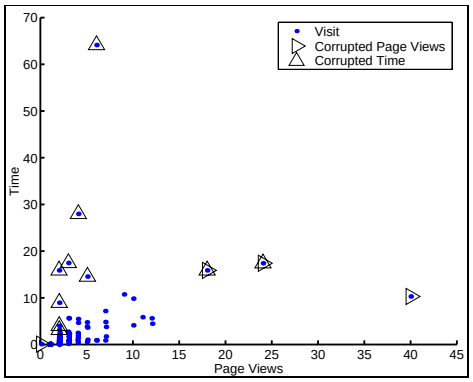

Figure 6. Web data with corruptions marked.

The results are also shown in Figure 6. Corrupted/Abnormal records are marked with one or two triangles that point along the axis in which the corruption was found. The results provide anomalies and explanations (direction of corruption) that correctly agree with intuition.

A word of caution should be added. The identified points may not truly fit the criterion of being anomalies; they might just be points that do not fit the clean model. For example, consider the case of four tight clusters that are modeled with 3 Gaussians. We would expect points from at least one cluster to be poorly accounted for by the Gaussians and thus be considered noise.

7 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 records. Further, it considers noise on the cell level, allowing it to use uncorrupted data from records with only a few corrupted attributes. We show that this approach can lead to benefits in classification, overall data accuracy, and model accuracy using both real world and artificial data.

Acknowledgements

Jeremy Kubica is supported by a grant from the Fannie and John Hertz Foundation. Andrew Moore supported by the National Science Foundation under Grant No. 0121671. The authors thank Dan Pelleg for his helpful comments and Theodore Kim for providing site counter data.

References

[1] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Knowledge Discovery and Data Mining*, pages 164–169, 1996.

[2] H. Attias, L. Deng, A. Acero, and J. 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.

[3] C. Blake and C. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

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

[5] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[6] D. Gamberger, N. Lavrač, and C. 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.

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

[8] I. Guyon, N. Matic, and V. Vapnik. Discovering informative patterns and data cleaning. In *Advances in Knowledge Discovery and Data Mining*, pages 181–203. 1996.

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

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

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

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

[13] V. T. Raisinghani. Cleaning methods in data warehousing. Seminar Report, 1999.

[14] B. Raj, M. L. Seltzer, and R. M. Stern. Reconstruction of damaged spectrographic features for robust speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*, 2000.

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

[16] S. Schwarm and S. Wolfman. Cleaning data with bayesian methods. 2000.

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