Cleaning Data with Bayesian Methods

Sarah Schwarm **Steve Wolfman**

{sarahs, wolf}@cs.washington.edu

March 16, 2000

Abstract

Data in many domains suffers corrupting processes. Our goal in this paper is to clean such data – *i.e.* reverse the effects of these corrupting processes. In part, we seek to produce data which supports the creation of better learners, but primarily, we want to produce data faithful to the "untampered, original" set.

Previous techniques have attempted to use learners to predict problems with class values in data sets. These techniques have now been applied not only to detect but also to correct errors in data. However, these techniques suffer several problems: they can only actually *correct* noise in the class attribute, they do not fully leverage dependencies among attributes, and they are inappropriate for data sets with no distinguished class attribute. More recent techniques have looked at characterizing errors in non-class attributes.

We use Bayesian techniques to take advantage of dependencies between attributes in a principled manner and to exploit expert knowledge of the relationships among the attributes. Our technique models the domain and noise generation process, rates its confidence that each instance is noisy (or asserts that the instance is clean), and suggests incremental corrections to nodes which appear to be corrupted.

1 Introduction

Large databases containing abundant varieties of data are proliferating in many domains. Any large set of data is almost guaranteed to contain errors from a variety of corrupting processes including data entry mistakes, measurement errors, missing values, and outliers.

The size of most real-world datasets requires that machine learning algorithms be used to extract meaningful patterns and knowledge from the data. The errors inherent in any large dataset may impair the performance of most learners. However, the size of the datasets make hand cleaning prohibitively expensive, even if we had access to a domain expert with enough knowledge to do it. Thus, we propose using automatic learning techniques not only to extract information from the data but to clean it first.

Previous techniques have used machine learning to predict errors in class values. This is effective if the domain is structured such that each data point has a set a features, one of which is the class, and if we know that all (or at least most) of the noise is in the class feature. However, this is not usually the case. The class value may be more accurate than the other features, in some cases. (If most of the errors are data entry mistakes, it is reasonable to think that the class might have fewer errors, since the people involved know that it is the "important" value and pays more attention to it.) Regardless of the distribution of noise, these techniques will not work at all in domains with no distinguished class value. Therefore, our technique strives to account for and take advantage of the relationships between *all* interdependent attributes and regularities in the noise generation process.

A second weakness with previous techniques is that they fail to exploit an important resource: expert human knowledge about the data generating and noise generating processes. In many domains, the problem with data cleaning is not the difficulty of correcting any individual but but simply be the sheer size of the dataset. To address this problem, database custodians create domain-specific heuristic rules which they apply to identify and sometimes fix problems with the data. Our system seeks to extract and leverage the expert knowledge behind these rule sets in a straightforward manner.

We use Bayesian networks specifically to address these two issues – interdependence among features and expert knowledge of the domain and the noise generation process itself. Bayes networks provide an interface in which a domain expert can easily construct a model of the structure of interactions among features in a problem. We can then superimpose a simple model of noise generation on the expert's model, and train the network on a small set of hand cleaned data (*along with* the corresponding corrupted data) to quantify the qualitative information supplied by the domain expert.

In this paper, we present our model of data cleaning. Then, we develop a methodology for detecting and correcting noise in a data set. Finally, we present an initial implemented system based on this model and describe the results of our experiments with this system.

2 Data Cleaning Model

The goal of data cleaning in this context is to identify and correct errors in the data. We envision a scenario in which a substantial amount of verifiable or otherwise correctable data is collected for any of a variety of eventual purposes. The custodians of the data set wish to clean it before its distribution in order to make the data as accurate as possible and maximize the benefit to clients of the database.

Unfortunately, even under the assumption that it is *possible* to clean the data, there is no reason to assume it is easy. Indeed, data cleaning is a tedious and time-consuming process. Therefore, we propose a model of data cleaning in which an automated system identifies and corrects noisy instances based on input from a human expert. The expert has a reasonably clear understanding of the process generating the data and is capable with some effort of reliably cleaning a small number of data points.

We require the expert to provide a model (in the form of a Bayesian network) of the basic structure of the data and clean a small percentage of the points by hand[1]. Our system rates its confidence of noise in each of the remaining, uncleaned data points. The system can then either present points which are likely to contain errors to an expert user for correction

[1]We set our experimental cleaning percentage to 5%, but this percentage would likely vary upward with the complexity of the domain and downward the size of the data set.

– focussing the expert's attention where it is most valuable – or it can modify the points according to its own predictions in order of its confidence in the order suggested by the list.

3 Methodology

Our system accepts as input a Bayesian network describing the underlying process which creates the data, the data set, and a small amount of data which has been cleaned along with the corresponding dirty points.

The network should include exactly one node for each feature in the data (and no more) and should have as reliable a structure as possible; however, initial descriptions of the conditional probability table are used only as seeds for the training process. We use each node in the model to represent a *true* (clean) value of the feature. We then modify the network to include an extra "shadow" node for each feature. Each shadow node has a single original node as its parent and represents the dirty version of that node.

To train this network, we combine the hand-cleaned instances and their corresponding dirty instances to form a training set with one value for each node (both true and shadow) in the network for each instance. Then, we use the Bayesian Knowledge Discoverer software [Ramoni and Sebastiani, 2000] to learn the most probable *a posteriori* network given our initial structure, a small weight on the initial conditional distributions of the network, and the observed data.

Finally, to clean a dirty instance, our system considers each feature in turn. When examining a given feature, we assume that all the other features are correct. We set all the "shadow" nodes to the values of that instance and also set all the clean values except the node for the feature under consideration to their values in the current instance[2]. Then the Bayesian network recalculates the probability distribution for each node, and we query the network to determine the most probable value for the cleaned version of the feature we are considering.

If the most probable value is the same as the dirty value, we report that the point is clean for that feature. Otherwise, we calculate a confidence in the new predicted value. Initially, we expected this confidence to be described by a combination of the probability that the current value of the feature is wrong ($\mathrm{P}(n_i \neq \mathrm{curr})$)[3] and the probability that the predicted value of the feature is correct ($\mathrm{P}(n_i = \mathrm{pred})$). Thus, we targeted the quantity $\mathrm{P}(n_i \neq \mathrm{curr} \wedge n_i = \mathrm{pred})$[4]. However, this produced a surprise:

$$\begin{aligned}\mathrm{P}(n_i \neq \mathrm{curr} \wedge n_i = \mathrm{pred}) &= \mathrm{P}(n_i \neq \mathrm{curr} | n_i = \mathrm{pred})\mathrm{P}(n_i = \mathrm{pred}) \\ &= \mathrm{P}(n_i = \mathrm{pred}) \qquad \text{given that curr} \neq \text{pred}\end{aligned}$$

There is no dependence in this equation on the probability that the current value of the point is incorrect! The problem is that, when assessing the points individually, the optimal

[2]Technically, each shadow node is independent of the rest of the network given its corresponding true node; however, we expect that future, more elaborate, noise models might invalidate this assumption.

[3]Note that these probabilities are all conditioned on the data described above: known values for all shadow nodes and for all true nodes but the current one.

[4]Examination of this equation itself revealed an assumption of zero-one loss. We address this issue in our conclusions and future work section, Section 6

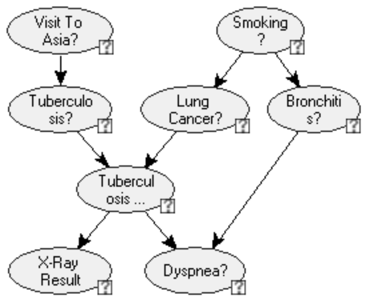

Figure 1: Bayes network for the Asia domain. Data was constructed directly from the network. The network for our system would add a shadow node as a child to each existing node.

action according to our model must be to change a feature to the predicted feature with the highest probability – a sensible choice. But, if there is no cost to this change, then there is no reason not to make the change in every case where the current value is not the highest probability value.

We solved this problem by setting our confidence to the expected benefit of altering the data point's value: $\mathrm{P}(n_i = \mathrm{pred}) - \mathrm{P}(n_i = \mathrm{curr})$. For instances with any feature predicted to be noisy, the maximum value of this measure over the features along with the corresponding feature number and predicted replacement value is reported in the output.

4 Experimental Results

We constructed two artificial domains to test our data cleaning methodology. In each one, we produced data sets of varying noise levels from 10% to 30% from initial clean data and used the system to attempt to recover the original data. For these initial experiments, we simply implemented *every* recommended correction on the data regardless of its confidence rating.

We made use of the Bayesian Knowledge Discoverer [Ramoni and Sebastiani, 2000] to create and train our networks and used the Smile/GeNIe library [Genie, 2000] to calculate cleaning confidences. All runs of the system took less than one minute of computer time[5].

We ran tests on datasets with 1000 instances total. Of these, 50 (5%) were used for training (and therefore supplied with both dirty *and* clean values) and the remaining 95% were used for testing.

4.1 Bayes Network Artificial Domain

Our first domain is designed specifically to test our technique. The data are produced from the underlying "Asia" model (in Figure 1) which is available with the Smile/GeNIe system. We simply sample the joint probability distribution induced by the Asia network to produce an arbitrary amount of data consistent with the network.

[5]We suspect that the rapid performance of the system was in part due to the many conditional independences induced by our evidence-heavy queries. This is among the reasons that we favored allowing only single, incremental changes to each data point.

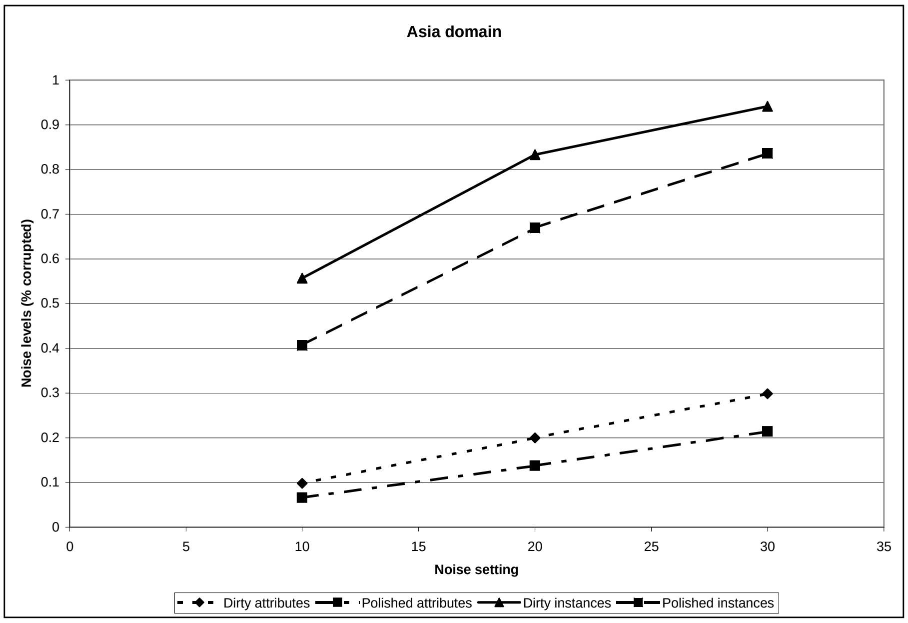

Figure 2: Results for the Asia domain varying the initial noise levels. Dirty noise percentages show the percent of the attributes/instances in the file with added noise. Polished noise percentages show the percent of attributes/instances that are noisy after running our polishing program.

To add noise to this network, we calculate a probability for each feature that we will reassign its value in a given instance to another value from the feature's domain. These values are tuned by a disparity factor. For zero disparity, the probabilities are uniform across all features. For higher disparities, we concentrate the noise in fewer features. As the disparity goes to infinity the noise concentrates entirely on a single feature. For values in the range of one to nine there tend to be between one and four features in our set of eight total features which account for the vast majority of the noise.

In our experiments, we supplied our system with the true underlying structure of the Bayes network but no conditional probability information. We tested datasets with noise levels of 10, 20 and 30% as shown in Figure 2 and also disparity levels of 0, 1, 2, and 3 at 10% disparity as shown in Figure 3. Clearly, our system provides substantial advantage, drastically reducing the number of noisy instances.

4.2 LED-24 Domain

We used a modified version of the LED-24 domain from the UCI Machine Learning repository. We introduced noise in the class attribute and modified the data generation program to output the corresponding clean values in addition to dirty values. Our model of the network is shown in Figure 4.

For this test, we again supplied our system with an accurate model of the domain. We tested datasets with 5, 10, 20 and 30% noise levels. Note that, given our model, our system is able to immediately disregard the effects of the random points in the LED-24 domain on all other points. We consider it a strength of the system that it trivially surmounts the

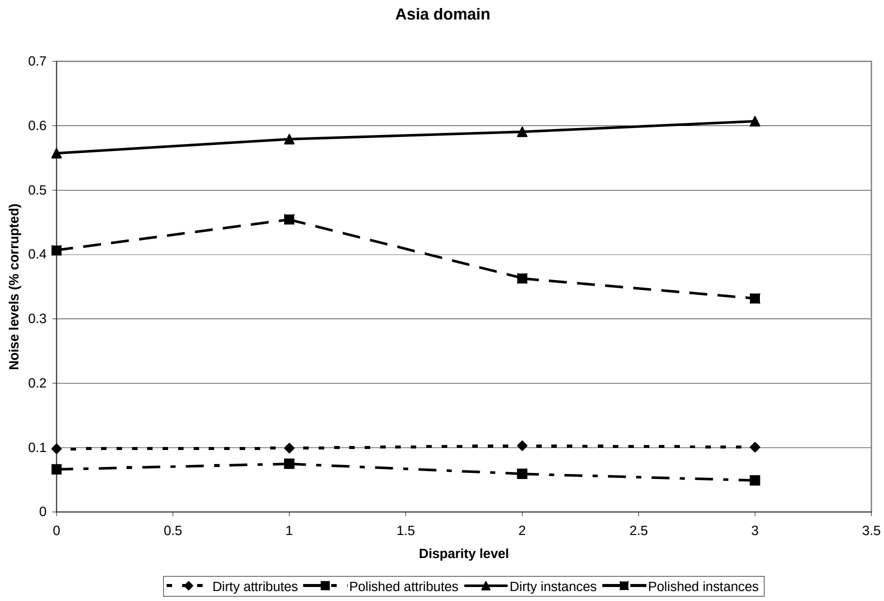

Figure 3: Results for the Asia domain varying the disparity level. Dirty noise percentages show the percent of the attributes/instances in the file with added noise. Polished noise percentages show the percent of attributes/instances that are noisy after running our polishing program.

pseudo-complexity of the LED domain. The results of the experiments are shown in Figure 5.

5 Related Work

Much previous work on data cleaning exists. Our work stems from the measurement of "surprise" to a learning system described by Guyon et al. [Guyon *et al.*, 1996]. More recent work on these lines extends Guyon's system to incorporate results from different types of learning systems[Brodley and Friedl, 1999] and exploit interdependence of features[Teng, 1999].

6 Conclusions

We have presented a new model for data cleaning which acknowledges and exploits the substantial domain-specific human expertise available for this problem. Guided by this model, we implemented a prototype system for automatically detecting and correcting noise in data.

The results of our preliminary experiments are promising. In all the tests we ran, we achieved a noticable improvement in the quality of the cleaned dataset. Modelling the data was fast and simple in the LED domain (where we constructed the model), and use of an existing model was trivial in the Asia domain.

Another important result of this initial work has been to suggest a wide variety of valuable extensions to, modifications of, and experiments on the system and model.

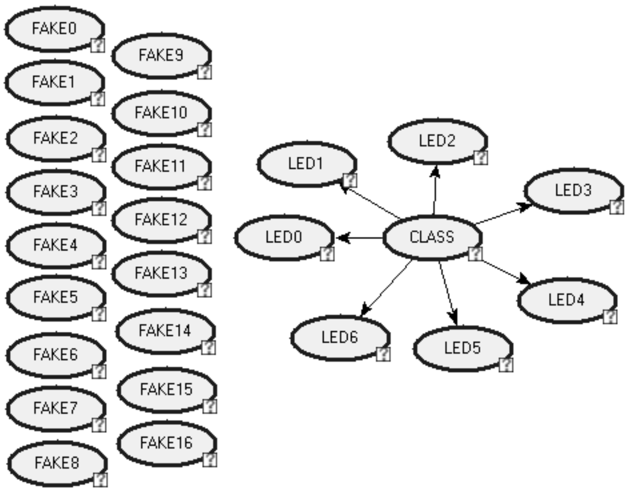

Figure 4: Bayes network for the LED domain. Data was constructed from the University of California at Irvine's LED-24 data-generating program. The network for our system would add a shadow node as a child to each existing node.

Currently, our system uses only the cleaned data points for learning and ignores a vast body of potential knowledge: the remaining dirty data points. Many other systems evaluate a given point based on all other (dirty) instances; such a technique could improve performance for our system. We envision including the dirty points in the learning process possibly by simply marking the clean features in each point as missing.

There are many aspects of the error correction process that remain to be explored. Our current system only fixes one error per instance. For higher noise percentages, we *must* fix more than one error per instance in order to completely clean the dataset. Making successive passes using the cleaned output of the previous pass as input to the current pass can allow us to fix additional errors in each instance. This amounts to a hillclimbing search for the optimal assignment to all features. Clearly, such a strategy may be susceptible to local optima; however, our initial tests showed some benefit on both of our test domains. Still, in all cases, the first pass provides the most benefit. More work is needed to determine the optimal number of successive passes. Along with these empirical tests, we should also investigate the theoretical properties of this hill-climbing search. Does it converge? If so, under what conditions does it converge to a local optimum? If not, would human intervention on one or more of the points that are diverging stabilize the process?

We would like to investigate predicting the most probable reassignement to all features' values simultaneously rather than predicting only single changes. Although this problem is computationally relatively difficult compared to the single feature problem, we have the advantage that we are only really concerned about relatively probable reassignments. Therefore, we suspect that a statistical approach (such as Monte Carlo sampling) will be appropriate for calculating a reassignment. Indeed, we might use only a small number of samples to judge the *inaccuracy* of the current assignment to the clean nodes given the dirty values and then continue to sample searching for a reassignment only if that probability is below some threshhold (*e.g.*, 50%).

We could expand the applicability of our technique by extending our metric to allow more

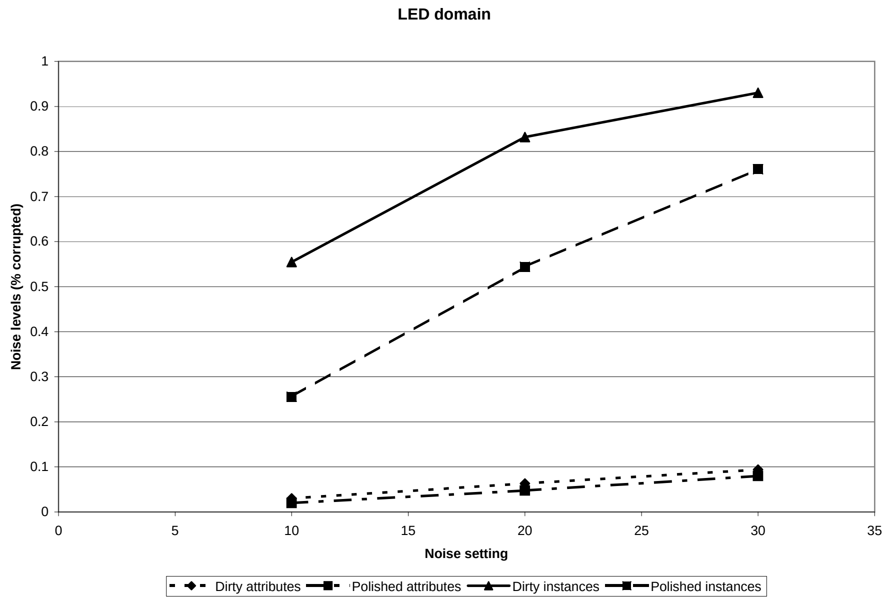

Figure 5: Results for the LED domain. Dirty noise percentages show the percent of the attributes/instances in the file with added noise. Polished noise percentages show the percent of attributes/instances that are noisy after running our polishing program.

complex loss functions than zero-on loss as a function of a more complex loss function. With a loss function $L(x, y)$ for the cost of assignment x if y is the true solution, and given the space of all possible assignments $\mathcal{A}$, our expected benefit from a reassignment would become $\Sigma_{x \in \mathcal{A}} \mathrm{P}(x)(L(\mathrm{curr}, x) - L(\mathrm{pred}, x))$. In the case of zero-one loss, this equation reduces to the one we used in our experiments.

We note also that the interesting aspects of the correction process do not cease when it is decided which points to correct. In fact, making the Bayes optimal decision in correcting a point will not necessarily give the best cleaning results. This is because if two points appear identical but one is actually a noisy manifestation of another instance, we can correct the wrong one but still obtain the full benefit of correction. Therefore, we can do better than the "Bayes optimal" decision when correcting a set of identical points by sampling from the probability distribution of reassignments for those points, leaving some the same and changing others. This technique relies on the interchangeability of points; so, it will reap no benefit in domains where each point is explicitly unique and must be avoided in domains where each point is implicitly unique (*i.e.*, where the order of the points is meaningful).

In our noise correction system, we could incorporate expert knowledge about the structure of the noise creation process. Currently, we assume that the creation of a noisy value is conditionally independent of all other values in the instance given the corresponding clean value. In some domains, this may be an invalid assumption, and a domain expert may be able to construct for our system a richer model of noise creation (which could then be refined through the same techniques described in this paper).

Finally, our current experiments use artificial data with artificial noise. Clearly, a vital next step is to run the system on real data. We would also like to design experiments which specifically target the value to the system of expert knowledge and interdependence of attributes.

References

[Brodley and Friedl, 1999] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[Genie, 2000] Genie and smile systems, 2000. The Decision Systems Laboratory of the University of Pittsburgh http://www.sis.pitt.edu/ dsl.

[Guyon *et al.*, 1996] Isabelle Guyon, Maticć Nada, and Vladimir Vapnik. Discovering Informative Patterns and Data Cleaning. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 181–204. AAAI Press, Menlo Park, CA, 1996.

[Ramoni and Sebastiani, 2000] Marco Ramoni and Paola Sebastiani. Bayesian knowledge discoverer system, 2000. Knowledge Media Institute, The Open University: http://kmi.open.ac.uk/projects/bkd/.

[Teng, 1999] Choh Man Teng. Correcting noisy data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.