Improving Mining Quality by Exploiting Data Dependency

Fang Chu Yizhou Wang Carlo Zaniolo D. Stott Parker

Computer Science Department, UCLA

Technical Report # TR050004

March 11, 2005

Abstract

The usefulness of the results produced by data mining methods can be critically impaired by several factors such as (1) low quality of data, including errors due to contamination, or incompleteness due to limited bandwidth for data acquisition, and (2) inadequacy of the data model for capturing complex probabilistic relationships in data. Fortunately, a wide spectrum of applications exhibit strong dependencies between data samples. For example, the readings of nearby sensors are generally correlated, and proteins interact with each other when performing crucial functions. Therefore, dependencies among data can be successfully exploited to remedy the problems mentioned above. In this paper, we propose a unified approach to improving mining quality using Markov networks as the data model to exploit local dependencies. Belief propagation is used to efficiently compute the marginal or maximum posterior probabilities, so as to clean the data, to infer missing values, or to improve the mining results from a model that ignores these dependencies. To illustrate the benefits and great generality of the technique, we present its application to three challenging problems: (i) cost-efficient sensor probing, (ii) enhancing protein function predictions, and (iii) sequence data denoising.

1 Introduction

The usefulness of knowledge models produced by data mining methods critically depends on two issues. (1) *Data quality*: Data mining tasks expect to have accurate and complete input data. But, the reality is that in many situations, data is contaminated, or is incomplete due to limited bandwidth for acquisition. (2) *Model adequacy*: Many data mining methods, for efficiency consideration or design limitation, use a model incapable of capturing rich relationships embedded in data. The mining results from an inadequate data model will generally need to be improved.

Fortunately, a wide spectrum of applications exhibit strong dependencies between data samples. For example, the readings of nearby sensors are correlated, and proteins interact with each other when performing crucial functions. Data dependency has not received sufficient attention in data mining research yet, but it can be exploited to remedy the problems mentioned above. We study this in several typical scenarios.

Low Data Quality Issue Many data mining methods are not designed to deal with noise or missing values; they take the data "as is" and simply deliver the best results obtainable by mining such imperfect data. In order to get more useful mining results, contaminated data needs to be cleaned, and missing values need to be inferred.

Data Contamination An example of data contamination is encountered in optical character recognition (OCR), a technique that translates pictures of characters into a machine readable encoding scheme. Current OCR algorithms often translate two adjacent letters " ff " into a "# " sign, or incur similar systematic errors.

In the OCR problem, the objective is not to ignore or discard noisy input, but to identify and correct the errors. This is doable because the errors are introduced according to certain patterns. The error patterns in OCR may be related to the shape of individual characters, the adjacency of characters, or illumination and positions. It is thus possible to correct a substantial number of errors with the aid of neighboring characters.

Data Incompleteness A typical scenario where data is incomplete is found in sensor networks where probing has to be minimized due to power restrictions, and thus data is incomplete or only partially up-to-date. Many queries ask for the minimum/maximum values among all sensor readings. For that, we need a cost-efficient way to infer such extrema while probing the sensors as little as possible.

The problem here is related to filling in missing attributes in data cleansing [8]. The latter basically learns a predictive model using available data, then uses that model to predict the missing values. The model train-

ing there does not consider data correlation. In the sensor problem, however, we can leverage the neighborhood relationship, as sensor readings are correlated if the sensors are geographically close. Even knowledge of far-away sensors helps, because that knowledge can be propagated via sensors deployed in between. By exploiting sensor correlation, unprobed sensors can be accurately inferred, and thus data quality can be improved.

Inadequate Data Model Issue Many well known mining tools are inadequate to model complex data relationships. For example, most classification algorithms, such as Naive Bayes and Decision Trees, approximate the posterior probability of hidden variables (usually class labels) by investigating on individual data features. These discriminative models fail to model the strong data dependencies or interactions.

Take protein function prediction as a concrete classification example. Proteins are known to interact with some others to perform functions, and these interactions connect genes to form a graph structure. If one chooses Naive Bayes or Decision Trees to predict unknown protein functions, he is basically confined to a tabular data model, and thus has lost rich information about interactions.

Markov networks, as a type of descriptive model, provide a convenient representation for structuring complex relationships, and thus a solution for handling probabilistic data dependency. In addition, efficient techniques are available to do inference on Markov networks, including the powerful *Belief Propagation* [20] algorithm. The power in modeling data dependency, together with the availability of efficient inference tools, makes Markov networks very useful data models. They have the potential to enhance mining results obtained from data whose data dependencies are underused.

Our Contribution The primary contribution of this paper is that we propose a unified approach to improving mining quality by considering data dependency extensively in data mining. We adopt Markov networks as the data model, and use belief propagation for efficient inference, so as to clean the data, to infer missing values, or to generally improve the mining results from a model that ignores data dependency. This paper may also contribute to data mining practice with our investigations on some real-life applications.

The primary contribution of this paper is that we propose a unified approach to improving mining quality by considering dependencies among the data intensively in data mining. We adopt Markov networks as the data model, and use belief propagation to efficiently compute the marginal or maximum posterior probability, so as to clean the data, to infer missing values, or to generally improve the mining results from a model that ignores data dependency.

This paper may also contribute to data mining practice with our investigations on several real-life applications. By exploiting data dependency in these application, clear improvements have been achieved in data quality and the usefulness of mining results.

Outline We describe Markov networks in the next section. Also discussed there are pairwise Markov networks, a special form of Markov network. Pairwise Markov networks not only model local dependency well, but also allow very efficient computation by belief propagation. We then address the three above-mentioned examples in sections 3, 4 and 5. We conclude the paper with related work and discussion in Section 6.

2 Markov Networks

Markov networks have been successfully applied to many problems in different fields, such as artificial intelligence [14], image analysis [17], turbo decoding [11] and condensed matter physics [1]. They have the potential to become very useful tools of data mining.

2.1 Graphical Representation The Markov network is naturally represented as an undirected graph $G = (V, E)$, where V is the vertex set having a one-to-one correspondence with the set of random variables $X = \{x_i\}$ to be modeled, and E is the undirected edge set, defining the neighborhood relationship among variables, indicating their local statistical dependencies. The local statistical dependencies suggest that the joint probability distribution on the whole graph can be factored into a product of local functions on cliques of the graph. A clique is a completely connected subgraphs (including singletons), denoted as X_C. This factorization is actually the most favorable property of Markov networks.

Let C be a set of vertex indices of a clique, and let $\mathcal{C}$ be the set of all such C. A *potential function* $\psi_{X_C}(x_C)$ is a function on the possible realization x_C of the clique X_C. Potential functions can be interpreted as "constraints" among vertices in a clique. They favor certain local configurations by assigning them a larger value.

The joint probability of a graph configuration $p(\{x\})$ can be factored into

$$\mathsf{P}(\{\mathsf{x}\}) = \frac{1}{\mathsf{Z}} \prod_{\mathsf{C} \in \mathcal{C}} \psi_{\mathsf{x_c}}(\mathsf{x_C}) \tag{2.1}$$

where Z is a normalizing constant:

$$\mathsf{Z} = \sum_{\{\mathsf{x}\}} \prod_{\mathsf{C} \in \mathcal{C}} \psi_{\mathsf{x_c}}(\mathsf{x_C})$$

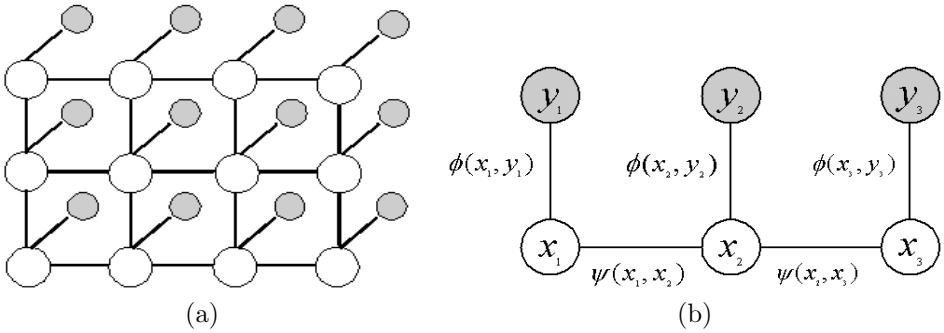


Figure 1: **Example of a Pairwise Markov Network. In (a), the white circles denote the random variables, and the shaded circles denote the external evidence. In (b), the potential functions $\phi()$ and $\psi()$ are showed.**

2.2 Pairwise Markov Networks Computing joint probabilities on cliques reduces computational complexity, but still, the computation may be difficult when cliques are large. In a category of problems where our interest involves only pairwise relationships among the samples, we can use use *pairwise Markov networks*. A pairwise Markov network defines potentials functions only on pairs of nodes that are connected by an edge.

In practical problems, we may observe some quantities of the underlying random variables $\{x_i\}$, denoted as $\{y_i\}$. The $\{y_i\}$ are often called evidence of the random variables. In the text denoising example discussed in Section 1, for example, the underlying segments of text are variables, while the segments in the noisy text we observe are evidence. These observed external evidence will be used to make inferences about values of the underlying variables. The statistical dependency between x_i and y_i is written as a joint compatibility function $\phi_i(x_i, y_i)$, which can be interpreted as "external potential" from the external field.

Another type of potential functions are defined between neighboring random variables. The compatibility function $\psi_{ij}(x_i, x_j)$ which captures the "internal binding" between two neighboring nodes i and j. An example of pairwise Markov networks is illustrated in Figure 1(a), where the white circles denote the random variables, and the shaded circles denote the evidence. Figure 1(b) shows the potential functions $\phi()$ and $\psi()$.

Using the pairwise potentials defined above and incorporating the external evidence, the overall joint probability of a graph configuration in Eq.(2.1) is approximated by

$$\mathsf{P}(\{\mathsf{x}\},\{\mathsf{y}\}) = \frac{1}{\mathsf{Z}} \prod_{(\mathsf{i},\mathsf{j})} \psi_{\mathsf{ij}}(\mathsf{x}_\mathsf{i}, \mathsf{x}_\mathsf{j}) \prod_{\mathsf{i}} \phi_\mathsf{i}(\mathsf{x}_\mathsf{i}, \mathsf{y}_\mathsf{i}) \qquad (2.2)$$

where Z is a normalization factor and the product over (i, j) is over all compatible neighbors.

2.3 Solving Markov Networks Solving a Markov network involves two phases:

- *The learning phase,* a phase that builds up the graph structure of the Markov network, and learns the two types of potential functions, $\phi()$'s and $\psi()$'s, from the training data.
- *The inference phase,* a phase that estimates the marginal posterior probabilities or the local maximum posterior probabilities for each random variable, such that the joint posterior probability is maximized.

In general learning is an application-dependent statistics collection process. It depends on specific applications to define the random variables, the neighborhood relationships and further the potential functions. We will look at the learning phase in detail with concrete applications in Sections 3-5.

The inference phase can be solved using a number of methods: simulated annealing [9], mean-field annealing [15], Markov Chain Monte Carlo [7], etc. These methods either take an unacceptably long time to converge, or make oversimplified assumptions such as total independence between variables. We choose to use the Belief Propagation method, which has a computation complexity proportional to the number of nodes in the network, assumes only local dependencies, and has proved to be effective on a broad range of Markov networks.

2.4 Inference by Belief Propagation Belief propagation (BP) is a powerful inference tool on Markov networks. It was pioneered by Judea Pearl [14] in belief networks without loops. For Markov chains and Markov networks without loops, BP is an exact inference method. Even for loopy networks, BP has been successfully used in a wide range of applications [11][12]. We give a short description of BP in this subsection.

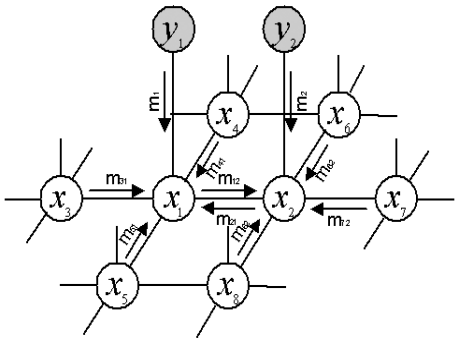


Figure 2: **Message passing in a Markov network. Messages are defined by Eqs.(2.3) or (2.4) under two types of rules, respectively.**

The BP algorithm iteratively propagates "messages" in the network. Messages are passed between neighboring nodes only, ensuring the local constraints, as shown in Figure 2. The message from node i to node j is denoted as $m_{ij}(x_j)$, which intuitively tells how likely node i thinks that node j is in state x_j. The message $m_{ij}(x_j)$ is a vector of the same dimensionality as x_j.

There are two types of message passing rules:

- *SUM-product rule*, that computes the marginal posterior probability.
- *MAX-product rule*, that computes the maximum a posterior probability.

For discrete variables, messages are updated using the SUM-product rule:

$$\mathsf{m}_{ij}^{t+1}(\mathsf{x_j}) = \sum_{\mathsf{x_i}} \phi_i(\mathsf{x_i}, \mathsf{y_i})\psi_{ij}(\mathsf{x_i}, \mathsf{x_j}) \prod_{\mathsf{k\in N(i), k\neq j}} \mathsf{m}_{ki}^{t}(\mathsf{x_i}) \quad (2.3)$$

or the MAX-product rule,

$$\mathsf{m}_{ij}^{t+1}(\mathsf{x_j}) = \max_{\mathsf{x_i}} \phi_i(\mathsf{x_i}, \mathsf{y_i})\psi_{ij}(\mathsf{x_i}, \mathsf{x_j}) \prod_{\mathsf{k\in N(i), k\neq j}} \mathsf{m}_{ki}^{t}(\mathsf{x_i}) \quad (2.4)$$

where $m_{ki}^t(x_i)$ is the message computed in the last iteration of BP, k runs over all neighbor nodes of i except node j.

BP is an iterative algorithm. When messages converge, the final belief $b(x_i)$ is computed. With the SUM-product rule, $b(x_i)$ approximates the marginal probability $p(x_i)$, defined to be proportional to the product of the local compatibility at node i ($\phi(x_i)$), and messages coming from all neighbors of node i:

$$\mathsf{b_i(x_i)_{SUM}} = \mathsf{x_i}\phi_i(\mathsf{x_i}, \mathsf{y_i}) \prod_{\mathsf{j\in N(i)}} \mathsf{m_{ji}(x_i)} \quad (2.5)$$

where $N(i)$ is the neighboring nodes of i.

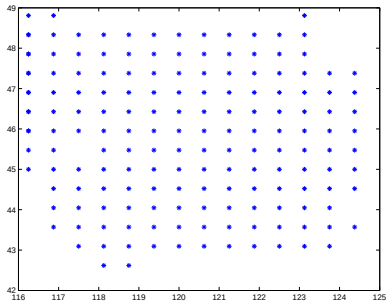


Figure 3: **Sensor site map in the states of Washington and Oregon.**

If using the MAX-product rule, $b(x_i)$ approximates the maximum a posterior probability:

$$\mathsf{b_i(x_i)_{MAX}} = \arg\max_{\mathsf{x_i}} \phi_i(\mathsf{x_i}, \mathsf{y_i}) \prod_{\mathsf{j\in N(i)}} \mathsf{m_{ji}(x_i)} \quad (2.6)$$

For more theoretical details of the belief propagation and its generalization, we refer the reader to [20].

3 Application I: Cost-Efficient Sensor Probing

In sensor networks, how to minimize communication is among the key research issues. The challenging problem is how to probe a small number of sensors, yet to effectively infer the unprobed sensors from the known. Cost-efficient sensor probing represents a category of problems where complete data is not available, but has to be compensated by inference.

Our approach here is to model a sensor network with a pairwise Markov network, and use BP to do inference. Each sensor is represented by a random variable in the Markov network. Sensor neighborhood relationships are determined by spatial positions. For example, one can specify a distance threshold so that sensors within the range are neighbors. Neighbors are connected by edges in the network.

In the rest of this section, we study a rainfall sensornet distributed over Washington and Oregon [13]. The sensor recordings were collected during 1949-1994. We use 167 sensor stations which have complete recordings during that period. The sensor site map is shown in Figure. 3.

3.1 Problem Description and Data Representation

The sensor recordings were collected in past decades over two states along the Pacific Northwest. Since rain is a seasonal phenomena, we split the data by week and build a Markov network for each week.

We need to design the potential functions $\phi_i(x_i, y_i)$ and $\psi_{ij}(x_i, x_j)$ in Eq. (2.2) in order to use belief propagation. One can use Gaussian or its variants to compute

the potential functions. But, in the sensornet we study, we find that the sensor readings are overwhelmed by zeroes, while non-zero values span a wide range. Clearly Gaussian is not a good choice for modeling this very skewed data. Neither are mixtures of gaussian, due to limited data. Instead, we prefer to use discrete sensor readings in the computation. The way we discretize data is given in section 3.3.

The $\phi()$ functions should tell how likely we observe a reading y_i for a given sensor x_i. It is natural to use the likelihood function:

$$\phi_{\mathsf{i}}(\mathsf{x_i}, \mathsf{y_i}) = \mathsf{P}(\mathsf{y_i}|\mathsf{x_i}) \tag{3.7}$$

The $\psi()$ functions specify the dependence of sensor x_j's reading on its neighbor x_i.

$$\psi_{\mathsf{ij}}(\mathsf{x_i}, \mathsf{x_j}) = \mathsf{P}(\mathsf{x_j}|\mathsf{x_i}) \tag{3.8}$$

3.2 Problem Formulation We give a theoretical analysis of the problem here. As we will see shortly, the problem fits well into the maximum a posterior (MAP) estimation on a Markov chain solvable by belief propagation.

Objective: MAP

Let X to be the collection of all underlying sensor readings, Y the collection of all probed sensors. Using Bayes' rule, the joint posterior probability of X given Y is:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{3.9}$$

Since $P(Y)$ is a constant over all possible X, we can simplify this problem of maximizing the posterior probability to be maximizing the joint probability

$$P(X, Y) = P(Y|X)P(X) \tag{3.10}$$

Eq.(3.10) is the objective function to be maximized, which is proportional to the maximum a posterior probability.

Likelihood

In a Markov network, the likelihood of the readings Y depends only on those variables they are directly connected to:

$$P(Y|X) = \prod_{i=1}^{m} P(y_i|x_i) \tag{3.11}$$

where m is the number of probed sensors.

Prior

Priors shall be defined to capture the constraints between neighboring sensor readings. By exploiting the Markov property of the sensors, we define the prior to involve only the first order neighborhood. Thus, the prior of a sensor is proportional to the product of the compatibility between all neighboring sensors:

$$P(X) \quad \propto \quad \prod_{(i,j)} P(x_j|x_i) \tag{3.12}$$

Solvable by BP

By replacing Eqs.(3.11) and (3.12) into the objective Eq.(3.10), we have the joint probability to be maximized:

$$P(X, Y) = \frac{1}{Z} \prod_{(i,j)} P(x_j|x_i) \prod_{i=1}^{N} P(y_i|x_i) \tag{3.13}$$

Looking back at the $\phi()$ and $\psi()$ functions we defined in Eqs.(3.7) and (3.8), we see that the objective function is of the form:

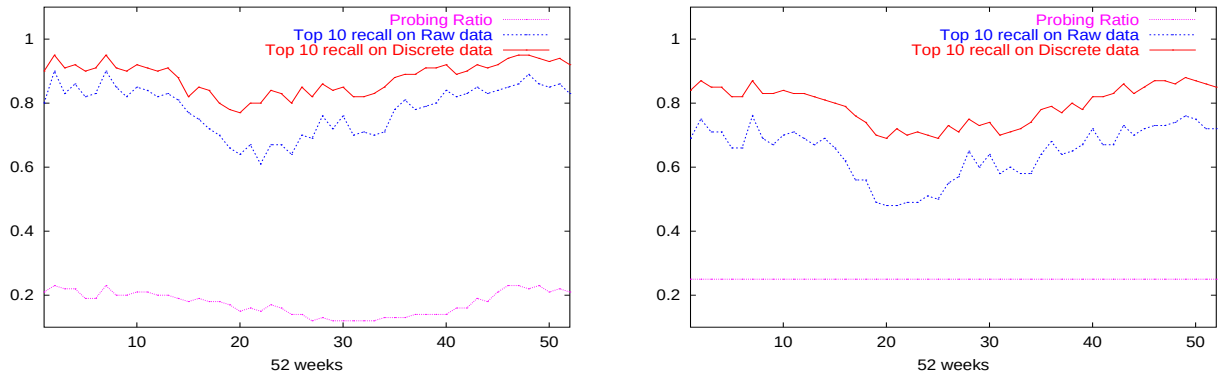
$$P(X, Y) = \frac{1}{Z} \prod_{(i,j)} \psi(x_i, x_j) \prod_{i=1}^{N} \phi(x_i, y_i) \tag{3.14}$$

where Z is a normalizing constant.

This is exactly the form in Eq.(2.2), where the joint probability over the pairwise Markov network is factorized into products of localized potential functions. Therefore, it is clear that the problem can be solved by belief propagation.

3.3 Learning and Inference The learning part is to find the $\phi()$ and $\psi()$ functions for each sensor, as defined in Eqs.(3.7) and (3.8). The learning is straightforward. We discretize the sensor readings in the past 46 years, use the first 30 years for training and the rest 16 years for testing. In the discrete space, we simply count the frequency of each value a sensor could possibly take, which is the $\phi()$, and the conditional frequencies of sensor values given its neighbors, which is the $\psi()$.

We use a simple discretization with a fixed number of bins, 11 bins in our case, for each sensor. The first bin is dedicated to zeroes, which consistently counts for over 50% of the populations. The 11 bins span the following ranges: $[0, 0]$, $[1, 5]$, $[6, 10]$, $[11, 30]$, $[31, 60]$, $[61, 100]$, $[101, 200]$, $[201, 400]$, $[401, 1000]$, $[1001, 1500]$, and$[1500, \infty)$. This very simple discretization method has been shown to work well in the sensor experiments. More elaborated techniques can be used which may further boost the performance, such as histogram equalization that gives balanced bin population with adaptive bin numbers.



(a) BP-based probing. (b) naive probing.

Figure 4: **Top-K recall rates vs. probing ratios. (a): results obtained by our BP-based probing; (b) by the naive probing. On average, BP-based approach probed 8% less, achieves 13.6% higher recall rate for raw values, and 7.7% higher recall rate for discrete values.**

For inference, belief propagation does not guarantee to give the exact maximum a posterior distribution, as there are loops in the Markov network. However, loopy belief propagation still gives satisfactory results, as we will see shortly.

3.4 Experimental Results We evaluate our approach using Top-K queries. A Top-K query asks for the K sensors with the highest values. It is not only a popular aggregation query that the sensor community is interested in, but also a good metric for probing strategies as the exact answer requires contacting all sensors.

We design a probing approach in which sensors are picked for probing based on their local maximum a posterior probability computed by belief propagation, as follows.

BP-based Probing:

1. Initialization: Compute the expected readings of sensors using the training data. Pick the top 20.
2. Probe the selected sensors.
3. True values acquired in step 2 become external evidence in the Markov network. Propagate beliefs with all evidence acquired so far.
4. Again, pick the top sensors with the highest expectations for further probing, but this time use the updated distributions to compute expectations. When there are ties, pick them all.
5. Iterate steps 2-4, until beliefs in the network converge.
6. Pick the top K with the highest expectations according to BP MAP estimation.

As a comparative baseline, we have also conducted experiments using a naive probing strategy as follows:

Naive Probing:

1. Compute the expectations of sensors. Pick the top 25% sensors.
2. Probe those selected sensors.
3. Pick the top K.

Performance of the two approaches is shown in Figure 4 (a) and (b), respectively. On each diagram, the bottom curve shows the probing ratio, and the two curves on the top show the recall rates for raw values and discrete values, respectively. We use the standard formula to compute recall rate, i.e.:

$$Recall = \frac{|S \bigcap T|}{|T|} \tag{3.15}$$

where S is the top-K sensor set returned, and T is the true top-K set.

Since the sensor readings are discretized in our experiments, we can compute S and T using raw values, or discrete values. Discrete recall demonstrates the effectiveness of BP, while raw recall may be of more interest for real application needs. As can be seen from Figure 4, raw recall is lower than discrete recall. This is due to error introduced in the discretization step. We expect raw recall to be improved when a more elaborated discretization technique is adopted.

It shows clearly in Figure 4 that BP-based approach outperforms the naive approach in terms of both recall rates, while requiring less probing. On average, the BP-based approach has a discrete recall of 88% and a raw recall of 78.2%, after probing only 17.5% sensors. The naive recall has a discrete recall of only 79.3%, a raw recall of only 64.6%, after probing 25% sensors.

The results shown in Figure 4 are obtained for $K = 10$. The relative performance remains the same for other values $K = 20, 30, 40$.

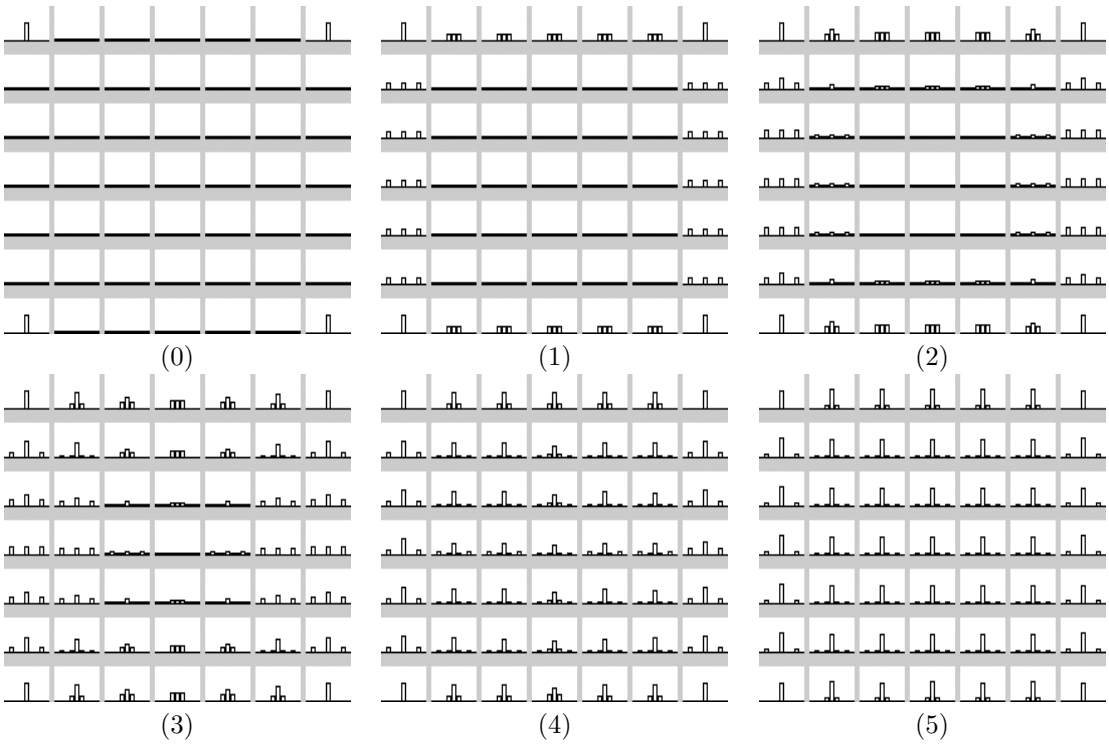


Figure 5: **Belief updates in 6 BP iterations((0) - (5)). Initially only the four sensors at the corners are probed. The strong beliefs of these four sensors are carried over by their neighbors to sensors throughout the network, causing beliefs of all sensors updated iteratively till convergence.**

3.5 How BP Works A closer look at the changing sensor beliefs during the iterations shows how belief propagation provides effective inference. We look at 49 sensors that form a 7×7 grid, each having the surrounding sensors (≤ 8) as its neighbors. Only the four sensors at the corners are probed. We use the $\psi()$ functions acquired by learning, but set $\phi()$ to be uniform. This is solely for demonstration purpose. (The original $\phi()$ is so skewed that BP converges too fast to demonstrate a moderate sized sequence of belief changes.) The beliefs are shown in Figure 5, one per iteration. In the first diagram, only the four corner sensors have an impulse at the true value, while all the others showing a flat distribution. But the probability histogram of each unprobed sensor grows notably sharper as BP iterates, showing how belief can grow stronger by receiving messages from neighbors.

This sensor probing on a small scale gives a sense of how effective belief propagation can be in Markov networks.

- From Figure 5, we can see that beliefs are able to propagate through the network via messages quickly. The messages of the four sensors at the corners are first passed to the nearby sites, then carried all the way to the central sites in just a few iterations.

- We can also see that the well informed nodes can help those less informed to build up their beliefs. Informally, we say a node is well informed or has stronger beliefs, if their belief distribution has a lower entropy. Figure 5 clearly shows that the four corner sensors pass strong beliefs to others to help them compute a good approximation of the posterior.

4 Application II: Enhancing Protein Function Predictions

Local data dependency can not only help infer missing values, as in the sensor example, but can also be exploited to enhance mining results. Many data mining methods, for efficiency consideration or design limitation, use a model incapable of capturing rich relationships embedded in data. Most discriminative models like Naive Bayes and SVM belong to this category. Predictions of these models can be improved, by exploiting local data dependency using Markov networks. The predictions are used as the likelihood proposal, and message passing between variables refines and reinforces the

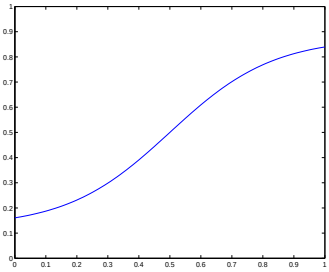


Figure 6: **Logistic curve that is used to blur the margin between the belief on two classes.**

beliefs. Next we show how to improve protein function predictions in this way.

4.1 Problem Description Proteins tend to localize in various parts of cells and interact with one another, in order to perform crucial functions. One task in the KDD Cup 2001 [3] is to predict protein functions. The training set contains 862 proteins with known functions, and the testing set includes 381 proteins. The interactions between proteins, including the testing genes, are given. Other information provided specifies a number of properties of individual proteins or genes that encodes the proteins. These include the chromosome on which the gene appears, phenotype of organisms with differences in this gene, etc.

Since information about individual proteins or genes are fixed features, it becomes crucial how to learn from interactions. According to the report of the cup organizers, most competitors organized data in relational tables, and employed algorithms that deal with tabular data. However, compared with tables, graphical models provide a much more natural representation for interacting genes. With a Markov network model, interactions can be modeled directly using edges, avoiding preparing a huge training table. Interacting genes can pass messages to each other, thus getting their beliefs refined together.

In the next of this section, we show a general way of enhancing a weak classifier by simply leveraging local dependency. The classifier we use is Naive Bayes, which is learned from the relational table. We build a Markov network, in which genes with interactions are connected as neighbors. The $\phi()$ function prediction comes from Naive Bayes, and the $\psi()$ are learned from gene interactions.

4.2 Learning Markov Network We separate the learning of each function, as focusing on one function a time is easier. There are 13 function categories, hence we build 13 Markov networks. To prepare the

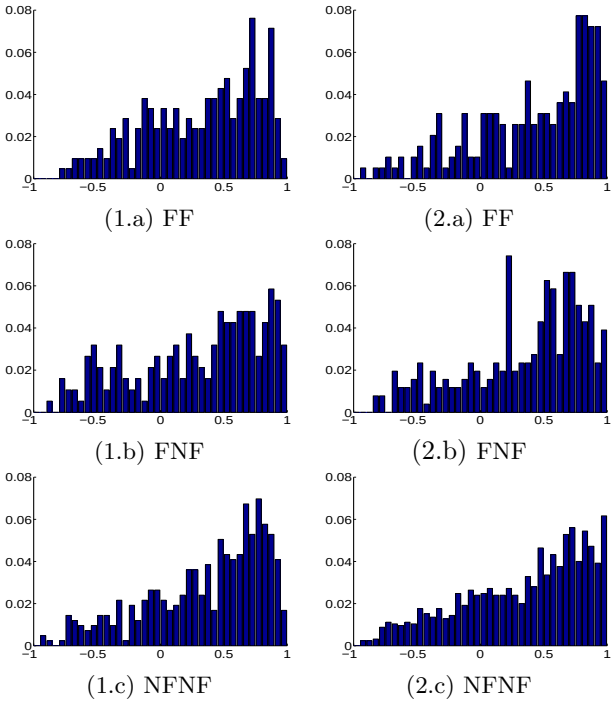


Figure 7: **Distribution of correlation values learned for two functions. left column function: cell growth, right column function: protein destination. In each column, the distributions from top to bottom are learned from group (a), (b) and (c), respectively.**

initial beliefs for a network, we first learn a Naive Bayes classifier, which output a probability vector $b_0()$, indicating how likely a gene will perform the function in question or not.

Each gene i maps to a binary variable x_i in the Markov network. First we design the $\phi()$ potentials for $\{x_i\}$. One can set the Naive Bayes prediction $b_0()$ to be $\phi()$. But this way the Naive Bayes classifier is over trusted, make it harder to correct the misclassifications. Instead, we adopt a generalized logistic function to blur the margin between the belief on two classes, yet still keeping the prediction decision.

$$f = \frac{a}{1 + e^{-\alpha(x-\beta)}} + b \tag{4.16}$$

In the experiments, we set $a = 0.75$, $b = 0.125$, $\alpha = 6$, and $\beta = 0.5$. The logistic curve is shown in Figure 6.

The $\psi()$ potentials are learned from protein interactions. Interactions are measured by the correlation between the expression levels of the two encoding genes. At first we tried to related the functions of two genes in a simple way: a positive correlation indicates that with

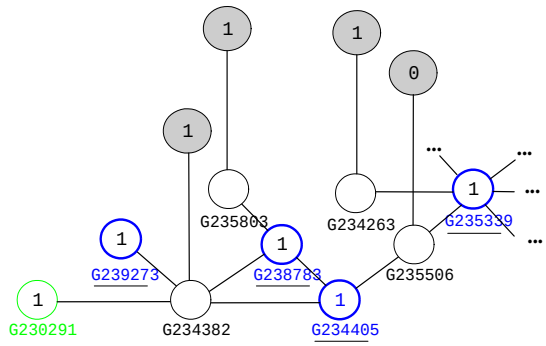


Figure 8: **A subgraph in which testing genes got correct class labels due to message passing.**

a fixed probability both or neither genes perform the function, while a negative correlation indicates that one and only one gene perform the function. This will leads to a simple fixed $\psi()$ function for all interacting genes. But, a close look at the interaction tells that 25% of the time this assumption is not true. In reality, sometimes two genes participating in the same function may be negatively correlated; a more influential phenomena is that genes may participate in several functions, hence the correlation is a combined observation involving multiple functions.

We decided to learn the distribution of correlation values for three types of interactions, separately: (a)FF: a group for protein pairs that both perform the function, (b)FNF: a group for pairs that one and only one performs the function, and (c)NFNF: a group for protein pairs that neither performs the function. Thus, the potential function $\psi_{i,j}$ defines how likely to observe a correlation value given for genes x_i and x_j, under different cases where x_i and x_j each has the function or not. In Figure 7, we show the distribution of correlation values learned for two functions. The left column is about a function related to cell growth, the right column is about a function related to protein destination. From top to bottom in each column, the distributions are learned from interaction group (a), (b) and (c), respectively. The figures show that correlation distributions differ among groups, and are specific to functions as well.

4.3 Experiments Naive Bayes does not perform well on this problem, because it does not model the gene interactions sufficiently, and thus cannot fully utilize the rich interaction information. Taking the average predictive accuracy of all classifiers, one per function, the overall accuracy of Naive Bayes is 88%. Belief propagation improves this to 90%.

To exemplify how misclassifications get corrected due to message passing, we show a subgraph of genes in Figure 8. The white circles represent genes(variables), and the shaded circles represent external evidence. Only training genes have corresponding external evidence. The 1's or 0's in the circles tell whether a gene has the function in question or not. For interested readers, we also put the gene ID below the circle. The subgraph contains four training genes and five testing genes. All these testing genes were misclassified by Naive Bayes. After receiving strong beliefs from their neighboring genes, four out of five testing genes were correctly classified. The other test gene 'G230291' was misclassified by both, but Naive Bayes predicted 0% for it to have the function (which is the truth), while belief propagation increased this belief to 25%.

We also evaluated our approach using the score function originally used in the 2001 KDD cup [3]. First we picked out all the functions we predicted for a gene. If more functions are predicted than the true number (which is actually the number of duplicates of that gene in the test table provided), we remove the ones with the smallest confidence. The final score is the ratio of correct predictions, including both positive and negative predictions. Our final score is 91.2%, close to the Cup winner's 93.6%. Although the winner scored reasonably high, they organized data in relational tables and didn't fully explore gene interactions. We expect that their method could perform better if integrated with our approach to exploit local dependencies between genes.

The Cup winner organized data in relational tables, which is not designed at all for complex relationships. To make up for this, they manually created new features, such as computing "neighbors" within k ($k > 1$) hops following neighbor links. Even so, these new features can only be treated the same as the other individual features. The rich relationship information in the original graph structure was lost. Graphical models, on the other hand, are natural models for complex relationships. Markov networks together with belief propagation provides a general and powerful modeling and inference tool on problems satisfying local constraints, such as protein function prediction.

5 Application III: Sequence Data Denoising

Sequences are ordered lists of elements, such as text strings, DNA sequences, or binary codes in channel transmission. This type of data often exhibits dependencies between adjacent elements. For example, there are rich dependencies embedded in English text. This sequence data can be modeled using Markov chains—a degenerate form of Markov networks.

Moreover, errors in sequence data often have neighborhood patterns. OCR discussed in Section 1 gives an

example where errors are related to the shapes of characters and to their relative positions. The mutation of a nucleotide is also influenced by its nearby bases. That the Markov property is satisfied by both the sequence data itself and by errors strongly suggests the applicability of belief propagation for sequence data denoising. Actually for Markov chains, belief propagation is theoretically guaranteed to give exact marginal or maximum a posterior probabilities.

In the rest of the section, we study a problem of correcting errors in noisy documents. While a simple problem, it exemplifies many basic characteristics in sequence data mining.

5.1 Problem Description and Data Representation A document is a text sequence consisting of characters from an alphabet, while a noisy document is the result from some recognizer with systemic errors. We split the sequence into small segments, each having n characters, and let neighboring segments overlap by m characters, $m < n$. We use a random variable $x_i = (x_i^{(1)}, \cdots, x_i^{(n)})$ to represent each underlying clean segment i. The corresponding observed segment in the noisy document is denoted as $y_i = (y_i^{(1)}, \cdots, y_i^{(n)})$. Each segment, except those that starts or ends the sequence, has a neighbor segment on either side.

Now we design the potential functions $\phi_i(x_i, y_i)$ and $\psi_{ij}(x_i, x_j)$. For $\phi()$, the definition should specify how likely we observe y_i given x_i. A natural choice is to define $\phi()$ to be a likelihood function

$$\phi_i(x_i, y_i) = P(y_i|x_i) \tag{5.17}$$

For a short segment, we can assume independence between characters. Thus, $\phi()$ can be written as

$$\phi_i(x_i, y_i) = \prod_{l=1}^{n} P(y_i^{(l)}|x_i^{(l)}) \tag{5.18}$$

For $\psi()$, the definition should specify how compatible two neighboring segments x_i and x_j are. Again, we can assume independence between the characters in the two segments, except for those in the overlapping part. Consider two overlapping characters, $x_i^{(k)}$ and $x_j^{(l)}$. If the probability is zero that $x_i^{(k)}$ will change to $x_j^{(l)}$ or vice versa, then the two segments, x_i and x_j, are incompatible. The resulting mutation probability of the overlapping segment quantifies the compatibility of two neighboring segments. Non-adjacent segments are incompatible. Formally, we define an asymmetric $\psi()$ function on x_i and x_j, when x_i is the left neighbor of x_j:

$$\psi_{ij}(x_i, x_j) = \prod_{l=1}^{m} P(x_j^{(l)}|x_i^{(n-m+l)}) \tag{5.19}$$

5.2 Learning and Inference The learning phase is to find the $\phi()$ and $\psi()$ functions. For this purpose, we build a mutation matrix M. Each matrix element $m(i, j)$ is the unconditional mutation probability from the i-th character to the j-th: $m(i, j) = P(ch_j|ch_i)$. This can be easily computed from the training set, which consists of pairs of clean and noisy documents.

We partition the clean document and the noisy documents in the same way. The $\phi()$ of each pair of clean and observed segments is given in Eq.(5.17), and the $\psi()$ of each pair of neighboring clean segments is given in Eq.(5.19).

In inference, a subproblem is to find candidate underlying segments for a given observed segment. One can enumerate all possible candidates using the mutation matrix. But this method not only can generate too many candidates, but also ignores valuable information in the training data: the possible combination of segments. We restrict the candidates to be the top matches among all training segments. When the number of matches is too small, we generate some extras using the transition matrix. By doing so, we actually explore the intra-segment constraints, which are fine details that the Markov chains cannot model, as they are on the scale of segments.

5.3 Experimental Results We choose two conference papers on the same topic: motion modeling. Both documents are distorted, using the probabilistic mutation rules in Table 1, to form pairs consisting of a clean document and a noisy document. One pair is used to train the potential functions, while the other is used for testing. For simplicity, we change all capitals into lower-case letters, replace all punctuation marks other than commas and periods into commas, and remove all figures, tables and equations. The transformed documents belongs to an alphabet of size 38 (consisting of 26 letters, 10 digits, a comma and a period).

A variety of distortion rules are used: unconditional mutation rules and k-order conditional mutation rules, $k = 1, 2, 3$. (A k-order conditional mutation depends on k neighbors on either side.) To compute the potential functions, all we need to learn is a 38 by 38 mutation matrix M for unconditional mutation rates only. Yet, we are able to catch and to correct most of the mutation errors, including the higher order conditional errors. In fact, the correction rates for conditional errors are even higher, as shown in Table 1. This is achieved by

rule	mutation prob.	# errors	% corrected
x → k	100%	56	91%
f → f	42%	-	-
f → d	30%	123	92%
f → z	28%	118	87%
th → th	48%	-	-
th → tn	52%	220	96%
se → se	36%	-	-
se → ue	18%	51	93%
se → le	25%	69	94%
se → ie	21%	58	95%
tio → tio	29%	-	-
tio → tho	20%	35	100%
tio → txo	20%	35	100%
tio → two	31%	57	98%
total words/errors: 3459/822		overall accuracy: 94%	

Table 1: **Distortion rules and error correction results. Columns 1 and 2 give the rule and mutation rate, respectively. Column 3 is the actual number of times a rule applies, and column 4 is the percentage corrected by BP inference.**

exploiting the Markov property and by passing local beliefs through the network using BP.

To help give an intuitive idea about how dependencies between text segments can be used effectively for error correction, we enclose a paragraph of distorted text here, followed by the corrected version. The misspelled words are underlined. We can see that most of the misspellings are corrected.

Distorted text:

introductxon. natural scenes contain rich stochastic mothon patterns which are characterized by the movement od a large number od distinguishable or indistinguishable elements, such as falling snow, zlock of birds, river waves, etc. tnele mothon patterns, called tektured, motion temporal tekture and dynamic tektures in the literature, cannot be analyzed by conventwonal optical zlow dields and have stimulated growing interests in both graphics and vision. in graphics, the objective is to render photorealistic video iequences, or non photorealistic but stylish cartoon animathons. both physics baued metnods such as partial didferential equatxons and image baled such as video tekture and volume tekture are studied to simulate dire, fluid, and gaseous phenomena. in vision, szummer and picard studied a spatial temporal auto regression star model, which is a causal gaussian markov random zield model.

Text after denoising:

introduction. natural scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of distinguishable or indistinguishable elements, such as falling snow, zlock of birds, river waves, etc. tnese motion patterns, called textured motion, temporal texture and dynamic tektures in the literature, cannot be analyzed by conventional optical flow fields, and have stimulated growing interests in both graphics and vision. in graphics, the objective is to render photorealistic video sequences, or non photorealistic but stylish cartoon animations. both physics based methods such as partial differential equations and image based such as video texture and volume texture are studied to simulate dire, fluid, and gaseous phenomena. in vision, szummer and picard studied a spatial temporal auto regression star model, which is a causal gaussian markov random field model.

6 Related Work and Discussions

Data dependency is present in a wide spectrum of applications. In this paper, we propose a unified approach that exploits data dependency to improve mining results, and we approach this goal from two directions: (1) improving quality of input data, such as by correcting contaminated data and by inferring missing values, and (2) improving mining results from a model that ignores data dependency.

Techniques for improving data quality proposed in the literature have addressed a wide range of problems caused by noise and missing data. For better information retrieval from text, data is usually filtered to remove noise defined by grammatical errors [16]. In data warehouses, there has been work on noisy class label and noisy attribute detection based on classification rules [21] [19], as well as learning from both labeled and unlabeled data by assigning pseudo-classes for the unlabeled data [2] using boosting ensembles. All this previous work has its own niche concerning data quality. Our work is more general in that it exploits local data constraints using Markov networks.

A pioneering work in sensor networks, the BBQ system [4] has studied the problem of cost-efficient probing. However, their method relies on a global multivariate Gaussian distribution. Global constraints are very strict assumptions, and are not appropriate in many practical scenarios.

The primary contribution of this paper is to propose a unified approach to improving mining quality by considering data dependency extensively in data mining. This paper may also contribute to data mining practice with our investigations on several real-life applications. By exploiting data dependency, clear improvements have been achieved in data quality and the usefulness of mining results.

Acknowledgement

We would like to thank Zhenyu Liu and Ka Cheung Sia for preparation of the sensor data and helpful discussions about the probing problem.

References

[1] S. Aji and R. McEliece. The generalized distributive law and free energy minimization. In *Proc. of the 39th Annual Allerton Conference on Communication, Control, and Computing*, 2001.

[2] K. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proc. of the 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 289-296*, 2002.

[3] J. Cheng, C. Hatzis, H. Hayashi, M.-A. Krogel, S. Morishita, D. Page, and J. Sese. Kdd cup 2001 report. In *SIGKDD Explorations, 3(2):47–64*, 2001.

[4] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *In Proc. of the 30th Int'l Conf. on Very Large Data Bases (VLDB 04)*, 2004.

[5] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. J. Wiley & Sons, New York, 1973.

[6] M. Elfeky, V. Verykios, and A. Elmagarmid. Tailor: a record linkage toolbox. In *In Proc. of the 18th Intl. Conf. on Data Engineering (ICDE 02)*, 2002.

[7] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. CRC Press, 1995.

[8] I. Guyon, N. Natic, and V. Vapnik. Discovering informative patterns and data cleansing. In *AAAI/MIT Press, pp. 181-203*, 1996.

[9] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. In *Science, vol. 220, no.4598*, 1983.

[10] M. Lee, H. Lu, T. Ling, and Y. Ko. Cleansing data for mining and warehousing. In *Int'l Conf. and Workshop on Database and Expert Systems Applications*, 1999.

[11] R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of pearl's 'belief propagation' algorithm. In *IEEE J. on Selected Areas in Communication, 16(2), pp. 140-152*, 1998.

[12] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empiricial study. In *Proc. Uncertainty in AI*, 1999.

[13] University of Washington. http://www.jisao.washington.edu/data_sets/widmann/.

[14] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann publishers, 1988.

[15] C. Peterson and J. Anderson. A mean-field theory learning algorithm for neural networks. In *Complex Systems, vol.1*, 1987.

[16] G. Salton and M. McGill. *Introduction to modern information retrieval*. McGraw Hill, 1983.

[17] R. Schultz and R. Stevenson. A bayesian approach to image expansion for improved definition. In *IEEE Trans. Image Processing, 3(3), pp. 233-242*, 1994.

[18] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[19] Y. Yang, X. Wu, and X. Zhu. Dealing with predictive-but-unpredictable attributes in noisy data sources. In *In Proc. of the 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 04)*, 2004.

[20] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. In *In Advances in Neural Information Processing Systems (NIPS), Vol 13, pp. 689-695*, 2000.

[21] X. Zhu, X. Wu, and Q. Chen. Eliminating class noise in large datasets. In *In Proc. of the 20th Int'l Conf. Machine Learning (ICML 03)*, 2003.