Under Review

Scaling Down EM Using Graph Based Models

Ian Davidson

Department of Computer Science, University of California, Davis, CA, USA.

Abstract.

The EM algorithm is used extensively in data mining problems that involve uncertainty in the form of missing values or latent variables. It was recently reported in KAIS as being one of the top 10 data mining algorithms. Typical applications include indoor map building, mixture models and image mining. However, as data mining applications extend to deployment beyond traditional desk top machines using typical probabilistic *parametric* models is infeasible. Resource constrained computational platforms such as hand-held GPS systems, small WiFi devices, and Sony Aibo robots have slow processors and no or limited floating point routines and hence are difficult to implement probabilistic formulations of EM on. In this paper we investigate the EM algorithm with undirected graphs as models. Constraints on the graph topology lead to different model classes and we explore several we believe have practical applications. We prove that the E-step reduces to the st-mincut problem which can be solved in polynomial time. For two of our model classes, we derive a non-heuristic M-step that can be carried out in polynomial time. These algorithms do **not** require floating point hardware and we demonstrate their performance on several classic problems.

1. Introduction and Motivation

Many important data mining problems can be cast as missing value or latent variable problems. Examples include constructing indoor maps from partially observed robot sensor data [3], learning first-order rules [14] and mixture models [17]. In map construction a robot sweeping its sensors left to right while walking down a corridor misses some location readings and must fill in these missing readings with values corresponding to say "wall" or "opening". Similarly with mixture models, the observed data variables are given to the algorithm and a latent variable (the cluster id) is said to be missing (unobserved). In both cases an algorithm is needed to use the observed data to fill in the missing values using some reasonable model. For example, EM could be used to repair the missing pixel values (shown as black) in Figure 1 or find the latent variables (clusters) in Figure 2.

Addressing problems containing missing values involves maximizing the **com-**

Fig. 1. A Problem With Missing Pixel Values That EM Can Address.

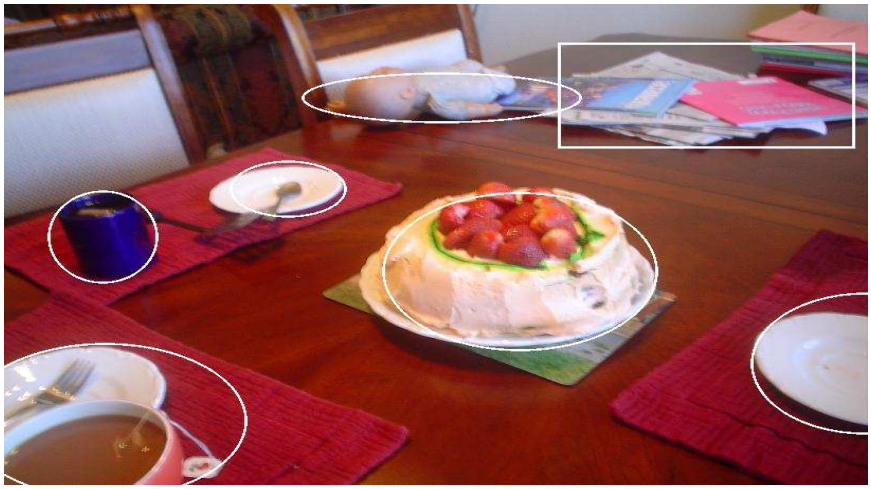

Fig. 2. A Problem with Latent Variables (the cluster id) That EM Can Address.

plete (missing and observed) likelihood which is typically intractable [9]. However, an algorithm, expectation maximization (EM) [16], is widely used and has produced useful results in many applications while being more efficient than other approaches, such as Gibbs sampling, at inferring missing vales [12]. Recently [22] the EM algorithm has been profiled by KAIS as one of the top ten data mining algorithms along with other classical algorithms such as Apriori and Support Vector Machines.

Since EM is rooted in the statistical literature it has been extensively applied using a variety of parametric models of the data. In this context, parametric model means a probabilistic model formulation characterized by the parameters of the probability distribution. For example, in the above applications [3, 17] a

Gaussian distribution is used with the parameters of course being the mean and variance. Before we introduce our graph-based models, it is worth noting that model classes provide the general form/structure of the relationship amongst the data with the EM algorithm finding the most probable parameters. Consider using EM for mixture models where the latent variable is the unobserved cluster label/id. A model space of a mixture of k multi-variate Gaussians enforces the data be modeled/divided as k sub-populations of Gaussians and EM finds the most probable parameters for such a division.

Unfortunately, many developing computationally limited platforms cannot readily run parametric versions of EM due to hardware limitations. There has been much discussion of scaling down algorithms to run on large sensor networks such inexpensive Berkeley motes [1] which even lack floating point hardware and not enough memory to implement the appropriate floating point routines. Furthermore, the inexpensive and noisy sensors return values that are best treated as binary (e.g. a temperature value > 25 Degrees Celsius is TRUE), and often return many missing or absurd values [11]. Similarly many computational devices are hand-held and portable and do not have the computational resources to run the EM algorithms. A recent innovation has been the move to hand-held devices running Windows CE such as global position systems (GPS) and hand-held computers which can access Wi-Fi connections (i.e. IPhones without the phone part) such as the Chumbys (www.chumby.com).

If mining applications built around EM are to be deployed on these computationally limited platforms, they must operate more efficiently both for real-time usage and battery life conservation. Approaches for speeding up EM do exist [17] but they typically involve pre-processing the data into some data structure which in itself would be time consuming and battery draining. An alternative approach is to transmit all observations back to a powerful server to perform the computations. However, this is also a significant power-draining activity and does not make use of the devices as a computational platform. In this paper we explore the approach of using graph based models so that mining applications based on EM can be implemented directly on the resource limited platforms.

Though there does exist some work on non-parametric EM, it is limited to 1) EM-style algorithms [5] that contain only an E-step and 2) model classes that are not useful for mining applications [20]. We explore using undirected graphs as non-parametric models and find that there exists a variety of constraints/topologies that can be imposed on the graphs to construct various model classes. Each graph/model has a vertex for each missing and observed value, with the edges being the model parameters. This simple approach can yield a variety of useful models.

Our work makes several pragmatic contributions:

– We show how to perform EM using graphs as models without the need for floating point hardware or routines. This allows efficient implementations of EM on resource limited hardware as floating point operations are not always available, take more memory and typically take significantly more time/power than integer operations[1] .

[1] Typically the Drystone benchmark (for integer and string operations) is three times as fast as the Whetstone benchmark (for floating point operations) for the same desktop CPU with large caches

– We illustrate that placing constraints on the graph topology is analogous to creating different model classes/spaces and identify several model classes.
– We present polynomial time E- and M-steps that **maximize** the likelihood. Hence, our techniques are **not** generalized EM (such as the Baum-Welch algorithm that have heuristic M-steps) and should have good convergence properties.

The remainder of this paper is organized as follows. In Section 2, we describe how the Hamiltonian energy function provides a probability function for undirected graphs. In Section 3, we describe four model classes that effectively impose constraints on the graph topology. Next (Section 4), we show how the general E-step for binary node values is facilitated by a polynomial time algorithm for the st-mincut problem. Then (Section 5), we derive efficient algorithms for the M-step of some model classes. Finally, we sketch an extension for multi-valued data and conclude our work.

Our earlier work [10] used a heuristic E-step which we improve upon in this paper (Section 4) and also used a simple nearest-neighbor model space and a different non-optimal heuristic M-step, thus leading to a generalized EM algorithm. Furthermore, the algorithm in our earlier work used an M-step that deleted **nodes** not **edges** as we do in this work. The deletion of nodes was very specific to sensor networks since it allows the turning off sensor nodes to conserve power and is not generalizable to other platforms.

2. A Graph-Based Model for Non-Parametric EM

In this section we discuss the general form of our non-parametric models and indicate how to measure the likelihood of a model. Given a set of observed (X) and missing (Y) values, our aim is to calculate the complete (missing and observed) maximum likelihood estimate $P(X, Y|\theta)$. We use classification EM (CEM) [2, 7, 19, 15] which maximizes the *classification complete likelihood* [7] and will allow computations free of floating point calculations. For instances with a single latent variable (Y) with $k = \{1, \ldots, K\}$ values, the classification complete likelihood [2] is given by:

$$P(X, Y|\theta) = \Pi_{x_i \in X} P(x_i|\theta) \times \Pi_{y_j \in Y} \Pi_k [P(y_j|\theta)]^{t_{j,k}} \quad (1)$$

where $t_{j,k}$ is the binary indicator classification variable, such that for each j, only one of the variables is set to 1 otherwise 0.

No tractable solution for maximizing the classification complete likelihood typically exists [9]. Intuitively, we can see why the problem is difficult. There are two sets of unknowns, the model parameters θ and the missing values (Y) that must be solved for. But both depend on one other: we use θ to predict the missing values and infer θ from the observed and missing values. However, the two steps of the CEM algorithm can be used iteratively to attempt to maximize the *expectation* of the log likelihood of the complete data given by $E_{P(Y|X,\theta')}[\log(P(Y, X|\theta))] = \sum_y \log(P(Y, X|\theta)).P(Y|X, \theta')$, where θ' is the previous guess/estimate of the model. The first step (the E-step) calculates the above expectation by setting the appropriate indicator variables to 1. This is

equivalent to setting the missing value to their most probable values. Given the filled in missing values, the second step (the M-step) calculates the value for θ that maximizes the expected complete data likelihood, i.e., $argmax_\theta P(Y, X|\theta)$. Then, one sets $\theta' = \theta$ and the two steps are repeated. This process monotonically increases the classification complete likelihood [7, 19], until a local maximum is found. We can now see how EM side-steps in the intractability issue. When two sets of dependent unknowns for a problem occur, the algorithm holds one set fixed (the model θ) and fills in the expectation of the other (the missing values Y) and then holds the missing values as fixed and estimates the most likely θ.

The use of EM requires the modeling of $Y|X, \theta$ and $Y, X|\theta$ which is typically done using some parametric model of the data such as a Gaussian. The parametric model plays the role of determining the relationship between the data. Instead of using a parametric model to determine the relationship between data, we use an undirected graph $G(V, E)$, where each value (observed, missing or latent variable) is a vertex and the *model parameters are the edges*. We begin our discussion by focusing on binary valued vertices for clarity and later discuss the straight-forward extensions to multi-valued vertices. The lattice spin-glass literature provides a well defined notion of a probability mass function for discrete physical systems using the Hamiltonian energy function [6]. To be consistent with the Ising spin-glass literature, graph nodes have the value $\{-1, +1\}$ when known and are set to '?' if missing. The Hamiltonian energy function ($H(G)$) and probability mass function ($P(G)$) of a particular configuration are given below. We need not explicitly calculate the maximum probability values as minimizing the Hamiltonian achieves the same goal.

Definition 2.1. Let $G(V, E)$ be an undirected graph. Suppose each node $v_i \in V$ is assigned a label $\ell_i \in \{+1, -1\}$.

$$H(G) = - \sum_{\{v_i, v_j\} \in E} \ell_i \, \ell_j \tag{2}$$

$$P(G) = \frac{e^{H(G)}}{\sum_{G'} e^{H(G')}} \tag{3}$$

Typically the summation over G' is over all configurations (possible values of ℓ_i) for a fixed set of edges, but it can be expanded to be over many graph topologies that comprise the model space. This sum is a normalizing constant that is typically ignored unless the purpose of the study is the simulation of the physical system. As can be seen, our probability measure is a function of the edges (E) in the graph and the node labels $(\ell_1, \ldots, \ell_n)$. As in the Ising spin-glass model, we sometimes refer to $H(G)$ as the **energy** associated with the given configuration. Note that the smaller the value of energy, the larger is the probability of the corresponding configuration. Also, as will be seen in Section 4, the smaller the number of conflicts between a node's value and its neighbors, the lower the energy of the graph. Our model, namely the edges of the graph, contains no continuous parameters, hence we use the term "non-parametric" to describe the model.

For the current graph (G'), the E-step will fill in the missing node values to minimize the Hamiltonian. The M-step must find the most probable model/graph (G) given the filled in values. The M-step can remove or add edges to obtain G, the most probable graph topology within a model class, given the vertex labels.

In this way we see the E-step as filling in the missing values and the M-step as changing the graph topology by adding or deleting edges. The M-step will also maintain a set of constraints on the graph topology give by the model class.

3. Definitions of Model Classes

In this section, we define some graph model classes that can be used with our EM algorithm. We note that the models are constraints on the graph topology but any arbitrary graph can be used to model the data within the limitations of the constraints. In all our graph model classes, the graphs are undirected without multi-edges or self-loops. The E-step discussed in the next section can be used for all these model classes. We derive polynomial time algorithms for the M-steps of two of the model classes; the remaining model classes are left for future work. It is worth noting that we focus on M-steps that find a model which **maximizes** the probability. This is desirable for fast convergence.

Model Class I – Lower Bound on Node Degree: This model class is characterized by a non-negative integer Δ. The requirement is that for each graph G in this model class and for each node v of G, the degree of v (i.e., the number of edges incident on v) is at least Δ. This model space is useful for spatial applications such as sensor networks.

Model Class II – Upper Bound on the Number of Connected Components: This model class is characterized by a positive integer r. The requirement is that each graph in this model class consists of **at most** r connected components. A connected component is a sub-graph on a subset of vertices such that there exists a path between each pair of vertices in that subset. The connected components are pairwise node disjoint. Using this model class for spatial data is equivalent to introducing a latent variable for each instance and maximizing the number of disjoint clusters (groups) to be at most r.

Model Class III – Upper Bound on Component Diameters: To define this model class, we need some graph theoretic definitions. Suppose G_i is a connected undirected graph. For any pair of nodes x and y in G_i, let $\delta_i(x, y)$ denote the number of edges in a shortest path between x and y in G_i. The **diameter** of G_i is given by $\max\{\delta_i(x, y) \ : \ x, y \in G_i\}$. If G_i consists of just a single node, then the diameter of G_i is defined to be zero. Model class III is characterized by a non-negative integer Γ. The requirement is that for every graph G in this model class and for every connected component G_i of G, the diameter of G_i is *at most* Γ. There is no constraint on the number of connected components. This model class could be useful for applications where prior some prior knowledge on the patterns are known. For example, nodes corresponded to pixels in an image, the value Γ would be the maximum width of an object.

Model Class IV – Lower Bound on Component Edge Density: Suppose $G_i(V_i, E_i)$ is a connected undirected graph. When G_i has no self-loops or multi-edges, the maximum number m_i of possible edges in G_i is given by $m_i = |V_i|(|V_i| - 1)/2$. The **density** of G_i is the ratio $|E_i|/m_i$. For convenience, when G_i consists of just a single node, we define the density of G_i to be 1. The density of G_i provides an indication of how close G_i is to the complete graph on $|V_i|$ vertices. Model class IV is characterized by the minimum density ρ, $0 < \rho \leq 1$. The requirement is the following. Suppose a graph G in this model

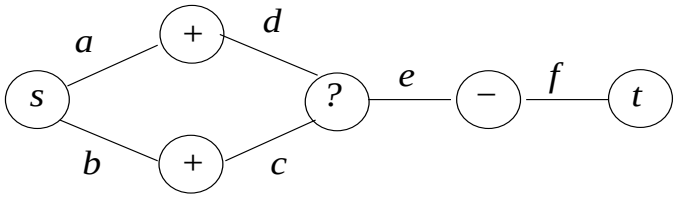

Fig. 3. A Simple Illustration Why the MinCut of G_1 Minimizes the Hamiltonian. Edges a, b and f are anchor edges.

class consists of connected components G_1, G_2, ..., G_t, for some $t \geq 1$. For each i, $1 \leq i \leq t$, the density of G_i is *at least* ρ.

Later, we present efficient algorithms for the M-step for Model Classes I and II defined above. Efficient M-steps for the remaining model classes are left for future work. Since a graph with minimum energy corresponds to a configuration of maximum probability, the focus of the M-step is to obtain a graph with the smallest amount of energy subject to the condition that the resulting graph satisfies the constraints imposed by the model class.

4. The E-Step

4.1. The E-Step - A Sketch

Given a graph $G(V, E)$ and a subset X of nodes that are the observed values, the goal of the E-step is to fill in the values for the nodes in $V - X$.

Calculating the most probable values for the missing value nodes is equivalent to minimizing the Hamiltonian $H(G)$ shown in Equation (2). Minimizing $H(G)$ is achieved by filling in the missing values so as to minimize the number of conflicts (differences) between neighbors. Since missing value nodes can be neighbors of other missing value nodes, filling in the missing values is not straightforward. Fortunately, this computation is tractable using the following approach. (For simplicity, we leave out several technical details in the following discussion. These details are provided in Section 4.4.)

To the graph G, we add two nodes (s and t), where s has the value $+1$ and t the value -1. All of the nodes in X whose value are $+1$ are connected to s and those whose value are -1 are connected to t. These new additional edges we shall call anchor edges. Note, the nodes with missing values $V - X$ are not connected to any anchors. This new graph is called G_1. All edges have unit weights, except that the anchor edges involving either s or t that have their weight as ∞. Then, a minimum weight edge cutset of G_1 is the minimum number of edges whose removal will create node-disjoint subgraphs: *at least one* positive subgraph where each node has the value $+1$ or '?' and *at least one* negative subgraph where each node has the value -1 or '?'. Determining the minimum weight edge cutset of a graph can be done in polynomial time (see for example, [21]). The subgraph ($+1$ or -1) that a missing value node is part of determines the value to be assigned to the node. Figure 3 shows the intuition behind why a minimum cut of the graph is needed. Clearly, in that figure, the missing value should be filled in as '+'. Removing edges marked c and d creates two appropriate subgraphs but it produces the wrong missing value; removing just the edge marked e corresponds

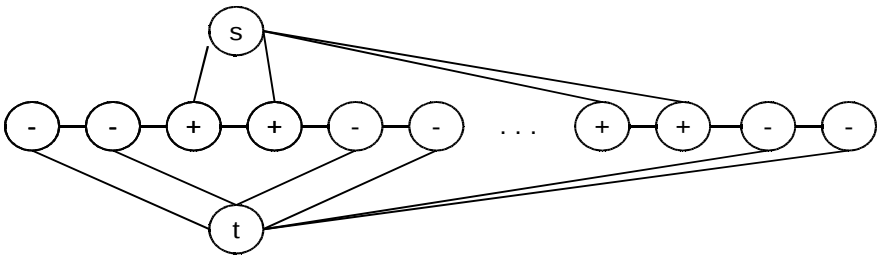

Fig. 4. A Illustration Of a MinCut of G_1 Which Will Produce Many Connected Components in G When Minimizing the Hamiltonian.

Input: An undirected graph $G(V, E)$; a subset P of nodes such that each node in P has been assigned a label from $\{+1, -1\}$. (These labels cannot be changed.)
Output: A label from $\{+1, -1\}$ for each node of $V - P$ such that $H(G)$ is minimized.
Algorithm:

1. Construct the auxiliary graph $G_1(V_1, E_1)$ from G.
2. Find an s-t edge cutset C^* of minimum weight in G_1.
3. Construct graph $G_2(V_2, E_2)$, where $V_2 = V_1$ and $E_2 = E_1 - C^*$, from G_1 by deleting the edges in C^*.
4. **for** each node $v \in V - P$ **do**

   if there is a path from s to v in G_2

   then Assign the label $+1$ to v

   else Assign the label -1 to v.

Fig. 5. Algorithm for the E-step

to a minimum cut and hence produces the correct result. This simple example masks a potentially complex behavior. The resultant edge removal could produce many connected components (disjoint sub-graphs) (if the edges to s and t are not considered). For example, consider Figure 4 where the min-cut will create many connected components.

Our algorithm shown in Figure 5 formalizes the above discussion. (Some details in the figure rely on definitions presented in Section 4.4.) The remainder of this section proves that this algorithm is indeed optimal and can be skipped on first reading without loss of continuity.

4.2. Derivation of the E-Step

Filling in the missing values with their most probable values is equivalent to calculating the *ground state* of the spin-glass model. The ground state involves

setting the missing values in such a way that the probability of the configuration (given by Equation (3)) is maximized. This, in turn, is equivalent to minimizing the energy value given by Equation (2). We now explain how this minimization problem can be solved in polynomial time.

4.3. Graph Theoretic Preliminaries

The model space considered in this paper consists of undirected graphs that are simple in the sense that they have no multi-edges or self-loops. Let $G(V, E)$ be an undirected graph. When each node $v_i \in V$ is assigned a value $\ell_i \in \{+1, -1\}$, the Hamiltonian function (or the **energy function**) $H(G)$ (Equation (3)) can be rewritten using the following definitions.

(a) Each edge $\{v_i, v_j\}$ such that $\ell_i \neq \ell_j$ is called a **conflict edge**.
(b) Each edge $\{v_i, v_j\}$ such that $\ell_i = \ell_j$ is called an **agreement edge**.

Lemma 4.1. Let N_c and N_a denote respectively the number of conflict and agreement edges in G. Then, $H(G) = N_c - N_a$. Alternatively, $H(G) = 2\,N_c - |E|$.

Proof: From the expression for $H(G)$ (Equation (2)), it is easy to see that each conflict edge contributes $+1$ to $H(G)$ and that each agreement edge contributes -1 to $H(G)$. Therefore, $H(G) = N_c - N_a$. Since each edge is either a conflict edge or an agreement edge, we have $|E| = N_c + N_a$. Therefore, $H(G)$ is also equal to $2\,N_c - |E|$. ∎

The following is an easy observation which will be used later.

Observation 4.1. Suppose $G(V, E)$ is an undirected graph where each node is labeled $+1$ or -1. If there is a path in G between a node labeled $+1$ and a node labeled -1, then the path has at least one conflict edge. ∎

4.4. An Efficient Algorithm for the E-Step

The E-step of the EM algorithm for filling in missing sensor node values solves the following combinatorial problem.

Minimum Energy Label Assignment (MELA)

Instance: An undirected graph $G(V, E)$ and a subset $P \subseteq V$ of "preassigned" nodes; that is, each node in P has been assigned a label from $\{+1, -1\}$.

Requirement: Assign a label from $\{+1, -1\}$ to each node in $V - P$ such that $H(G)$ is minimized.

It is assumed that in G, there is a path from each node in $V - P$ (i.e., each node with a missing value) to a node in P (i.e., a node which has a preassigned value from $\{+1, -1\}$). This assumption enables the E-step to assign values to missing nodes in an unambiguous fashion.

Our algorithm for the MELA problem relies on a transformation to the problem of finding a **minimum weight** $s-t$ **edge cut** in an undirected graph. The definition of such an edge cut is given below.

Definition 4.1. Let $G(V, E)$ be an undirected graph with a non-negative weight $w(e)$ for each edge $e \in E$. Let s and t be two distinct vertices in V. An s-t **edge**

cutset for G is a subset $E' \subseteq E$ such that in the graph $G'(V, E - E')$, there is no path between s and t. A **minimum weight s-t edge cutset** for G is an edge cutset whose total weight is minimum.

The following well known result shows that minimum weight edge cutsets can be found efficiently (see for example [21]).

Theorem 4.1. Given an undirected graph $G(V, E)$ with a non-negative weight $w(e)$ for each edge $e \in E$ and two distinct vertices s and t in V, a minimum weight s-t edge cutset for G can be computed in $O(|E| + |V| \log |V|)$ time. ∎

Recall that in the MELA problem, the nodes in the set $P \subseteq V$ have pre-assigned labels which cannot be changed. Throughout this section, we will use E^P denote the set of edges where each edge has both of its endpoints in P. Let N_c^P and N_a^P denote the number of conflict and agreement edges in E^P. (Thus, $|E^P| = N_c^P + N_a^P$.) The contribution H^P of the edges in E^P to the Hamiltonian of the graph G is therefore given by $H^P = N_c^P - N_a^P$. Note that no matter how labels are assigned to the nodes in $V - P$, the edges in E^P will always contribute H^P to the value of $H(G)$.

We now discuss how the MELA problem can be solved efficiently. Let $G(V, E)$ and $P \subseteq V$ denote the given instance of the MELA problem. Consider the auxiliary edge weighted graph $G_1(V_1, E_1)$ constructed from G as follows.

(a) $V_1 = V \cup \{s, t\}$, where s and t are two new nodes (i.e., $s \notin V$ and $t \notin V$).
(b) $E_1 = (E - E^P) \cup E_s \cup E_t$, where $E_s = \{\{s, v_i\} : v_i \in P \text{ and } \ell_i = +1\}$, and $E_t = \{\{t, v_i\} : v_i \in P \text{ and } \ell_i = -1\}$.
(c) For each edge $e \in E_1$, the weight of e, denoted by $w(e)$ is chosen as follows: if $e \in E$, then $w(e) = 1$; otherwise, $w(e) = \infty$.

We note that the auxiliary graph G_1 has a trivial s-t edge cutset of weight $|E - E^P|$. Thus, no minimum weight s-t edge cutset of G_1 can use any of the edges incident on the nodes s and t. In other words, any minimum weight s-t edge cutset of G_1 is a subset of $E - E^P$. The following lemma shows the role played by auxiliary graph in solving the MELA problem.

Lemma 4.2. Let $G(V, E)$ and $P \subseteq V$ constitute a given instance of the MELA problem. Let $H^*(G)$ denote the minimum value of the Hamiltonian function over all assignments of labels to the nodes in $V - P$. Let $G_1(V_1, E_1)$ denote the auxiliary graph of G constructed as discussed above and let W_1^* denote the minimum weight of an s-t edge cutset in G_1. Then, $H^*(G) = H^P + 2\,W_1^* - |E - E^P|$, where H^P is the contribution due to the edges in E^P.

Proof: We prove that result in two parts.

Part 1: Here, we prove that $H^*(G) \geq H^P + 2\,W_1^* - |E - E^P|$. Consider an assignment of labels from $\{+1, -1\}$ to the nodes in $V - P$ such that the value of $H(G)$ is equal to $H^*(G)$. Let C denote the set of all the conflict edges from $E - E^P$ in the resulting assignment. As mentioned earlier, the edges in E^P contribute H^P to $H(G)$, regardless of the label assignment to the nodes in $V - P$. From Lemma 4.1, the contribution to the Hamiltonian due to the edges in $E - E^P$ is $2|C| - |E - E^P|$. Therefore, $H^*(G) = H^P + 2|C| - |E - E^P|$. Now, we have the following claim.

Claim: C is an s-t edge cutset for G_1.

Proof of Claim: Suppose C is not an s-t edge cutset for G_1. Then, there is a path from s to t in the graph $G_2(V_1, E_1 - C)$. In this path, let x be the node that is adjacent to s and let y be the node that is adjacent to t. Thus, the label of x is $+1$ and that of y is -1. Hence, by Observation 4.1, there is a conflict edge in this path. By our construction of graph G_1, this conflict edge is from the edge set $E - E^P$. This contradicts the assumption that C contains all the conflict edges from $E - E^P$, and the claim follows.

In view of the claim, G_1 has an s-t edge cutset of weight at most $|C|$. Since W_1^* is the minimum weight of an s-t edge cutset of G_1, we have $|C| \geq W_1^*$. Therefore, $H^*(G) = H^P + 2\,|C| - |E - E^P| \geq H^P + 2\,W_1^* - |E - E^P|$. This completes the proof of Part 1.

Part 2: Here, we prove that $H^*(G) \leq H^P + 2\,W_1^* - |E - E^P|$. This is done by finding an assignment of labels to the nodes in $V - P$ such that the value of the Hamiltonian for the resulting assignment is at most $H^P + 2\,W_1^* - |E - E^P|$.

Consider algorithm in Figure 5. Using the assumption that there is a path in G from each node in $V - P$ to a node in P, it is easy to see that Step 4 of the algorithm assigns a label from $\{+1, -1\}$ to each node of $V - P$. Further, the assignment ensures that the only conflict edges from $E - E^P$ in the resulting assignment are those in C^*. Therefore, by Lemma 4.1, the value of the Hamiltonian function $H(G)$ for this assignment of labels to the nodes in $V - P$ is given by $H(G) = H^P + 2\,|C^*| - |E - E^P|$. Since C^* is a minimum weight s-t edge cutset, we have $|C^*| = W_1^*$. Therefore, $H(G) = H^P + 2\,W_1^* - |E - E^P|$. Since there is an assignment of labels to the nodes in $V - P$ such that the Hamiltonian function of G has a value of $H^P + 2\,W_1^* - |E - E^P|$, it follows that $H^*(G) \leq 2\,W_1^* - |E - E^P|$. This completes the proof of Part 2 as well as that of the lemma. ∎

A direct consequence of the above lemma is that the algorithm in Figure 5 computes an optimal solution to the MELA problem. The running time of the algorithm is dominated by Step 2, where a minimum weight s-t edge cutset of G_1 is constructed. As mentioned in Theorem 4.1, this step can be carried out in $O(|E| + |V| \log |V|)$ time. The following theorem summarizes the above discussion.

Theorem 4.2. The E-step of the EM algorithm for filling in the missing sensor values can be solved in $O(|E| + |V| \log |V|)$ time, where $|V|$ is the number of nodes and $|E|$ is the number of edges in the given sensor network. ∎

5. The M-Step

Unlike the E-step where one algorithm sufficed, the M-step that is required to maximize the likelihood varies from model space to model space. As can be seen from the discussion in Section 4, the energy of a graph can be reduced by deleting conflict edges. Therefore, the algorithms for the M-step considered in this section focus on deleting a maximum number of conflict edges. Thus, the edge set of the graph that is output by the M-step is a subset of the edge set of the input graph. Formally, the output of each M-step is an **edge subgraph** of the input graph.

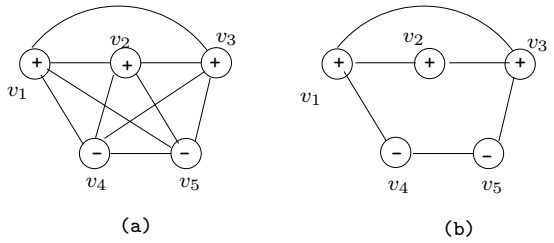

Fig. 6. Example of M-step for Model Class I

5.1. M-Step Algorithm for Model Class I

Here, we consider Model Class I, where there is a lower bound Δ on the degree of each node of the graph that is produced by the M-step. A formal statement of the problem solved by the M-step for this model class is as follows.

Graph with Lower Bound on Node Degree (GLBND)

Instance: An undirected graph $G(V, E)$, an integer $\Delta \leq |V| - 1$ such that each node has a degree of at least Δ, a label $\ell_i \in \{+1, -1\}$ for each node $v_i \in V$.

Requirement: Find an edge subgraph $G'(V, E')$ of G such that each node in G' has degree at least Δ and the energy of G' is a minimum among all edge subgraphs satisfying the degree constraint.

An example to illustrate the M-step for Model Class I is shown in Figure 6. The graph in Figure 6(a) has 6 conflict edges and 4 agreement edges for an energy value of $+2$. This graph represents the input to the M-step. Suppose the value of Δ is 2. The graph in Figure 6(b) is the output of the M-step. This graph has 2 conflict edges and 4 agreement edges for an energy value of -2. Note that each node in the output graph has a degree of at least 2.

We now discuss how the GLBND problem can be solved efficiently. We assume that in the graph which is given as input to the M-step, the degree of each node is at least Δ; if this condition is not satisfied, there is no solution to the problem under the chosen model class. Our algorithm is based on an efficient reduction of the GLBND problem to the following problem, which can be solved in polynomial time (see [13, 18]).

Minimum Cost Degree-Constrained Subgraph Problem (MCDCS)

Instance: An undirected graph $G(V, E)$, a cost $\alpha(e)$ for each edge $e \in E$, an integer $\Delta \leq |V| - 1$.

Requirement: Find an edge subgraph $G'(V, E')$ of G such that each node in G' has degree at least Δ and the cost of G' (i.e., the sum of the costs of the edges in E') is a minimum among all edge subgraphs satisfying the degree constraint.

References [13, 18] consider a more general version of the MCDCS problem, where both lower and upper bounds on node degrees are permitted, and show that the general version can also be solved in polynomial time. The simpler version of the MCDCS problem, as defined above, is adequate for our purposes.

An algorithm for the GLBND problem is given in Figure 7. Since the algorithm

1. Let $G(V, E)$ be the graph which forms a part of the GLBND instance. Construct an instance of the MCDCS problem as follows.
   (a) The undirected graph $G_1(V_1, E_1)$ for the MCDCS instance is an isomorphic copy of G.
   (b) For each conflict edge e of G, the corresponding edge e_1 in G_1 is assigned a cost of 1. For each agreement edge e of G, the corresponding edge e_1 in G_1 is assigned a cost of 0.
   (c) The lower bound on the degree of each node is set to Δ.
2. Find an optimal solution $G'_1(V_1, E'_1)$ to the MCDCS instance constructed in Step 1.
3. Output the graph $G'(V, E')$, where E' is the set of edges edges of G that correspond to the edges of G'_1.

Fig. 7. An Efficient Algorithm for the M-step under Model Class I

for the MCDCS problem (Step 2) runs in polynomial time, it is clear that the algorithm in Figure 7 also runs in polynomial time. The following lemma proves the correctness of the algorithm.

Lemma 5.1. In the subgraph $G'(V, E')$ output by the algorithm in Figure 7, each node has a degree of at least Δ. Further, the energy of G' is the smallest among all subgraphs satisfying the degree constraint.

Proof: In the graph $G(V, E)$ which was provided as input to the algorithm, each node has a degree of at least Δ. Therefore, there is a feasible solution to the MCDCS problem. Hence, the graph G' produced by the algorithm for the MCDCS problem also satisfies the degree constraint.

To see that G' has the lowest energy value among all the edge graphs of G satisfying the degree constraint, we note that Step 2 of the algorithm constructs an optimal solution to the MCDCS instance. Since the agreement edges have a cost of zero and adding edges cannot violate the lower bound on the degrees of nodes, we may assume that the algorithm for the MCDCS problem obtained the solution by deleting conflict edges only. Therefore, if there is an edge subgraph of G that satisfies the degree constraint and has a lower energy value than G', then there would be a lower cost solution for the MCDCS instance. This is a contradiction, and the lemma follows. ■

The following theorem summarizes the main result of this subsection.

Theorem 5.1. For the model class consisting of graphs with a specified lower bound on the degree of each node, the M-step can be carried out in polynomial time. ■

5.2. M-Step Algorithm for Model Class II

Here, we consider Model Class II, where each graph consists of at most r connected components (CCs), for a given integer r. The reader should bear in mind

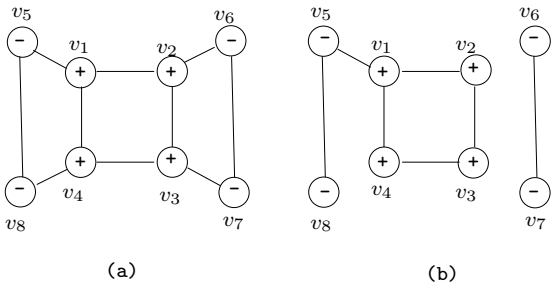

Fig. 8. Example of M-step for Model Class II

that the connected components are pairwise node disjoint. A formal statement of the problem solved by the M-step for this model class is as follows.

Graph with an Upper Bound on the Number of Connected Components (GUBCC)

Instance: An integer r, an undirected graph $G(V, E)$ consisting of at most r CCs, where each node v_i has a label $\ell_i \in \{+1, -1\}$.

Requirement: Find an edge subgraph $G'(V, E')$ of G such that G' also consists of at most r CCs and the energy of G' is a minimum among all edge subgraphs with at most r CCs.

An example to illustrate the M-step for Model Class II is shown in Figure 8. The graph in Figure 8(a) has 4 conflict edges and 6 agreement edges for an energy value of -2. This graph represents the input to the M-step. Suppose the value of r is 2. The graph in Figure 8(b), which has 2 connected components, is the output of the M-step. This graph has 1 conflict edge and 6 agreement edges for an energy value of -5.

We now discuss how the GUBCC problem can be solved efficiently. As in Section 5.1, we assume that the graph which is given as input to the M-step consists of at most r CCs. If this condition is not satisfied, there is no solution to the problem under the chosen model class since the approach to minimize energy involves deletion of edges. The idea behind our polynomial algorithm for the M-step for this model class is quite simple. We start by deleting all the conflict edges from the input graph. (This is done because deleting conflict edges reduces energy.) Let t denote the number of connected components after this deletion step. If $t \leq r$, then we clearly we have a solution with minimum energy value (which is equal to $-N_a$, where N_a is the number of agreement edges). Otherwise, we must reinsert $t - r$ of the conflict edges so as to reduce the number of connected components to r. Adding conflict edges to reduce the number of connected components to r can be done in a manner similar to Kruskal's algorithm for constructing a minimum spanning tree of a connected graph [8]. The resulting graph has the smallest energy value since to reduce the number of connected components from t to r, at least $t-r$ conflict edges must be reinserted. The steps of the algorithm are shown in Figure 9. The correctness of the algorithm follows from the above discussion. From an examination of Figure 9, it can be seen that the algorithm runs in polynomial time. The following theorem summarizes the main result of this subsection.

1. Let $G(V, E)$ be the graph which forms a part of the GUBCC instance. Delete all conflict edges of G to form graph $G_1(V, E_1)$. Let t denote the number of connected components of G_1.
2. **if** $t \leq r$

   then Output $G_1(V, E_1)$ as the solution.

   else

   (a) Add $t - r$ of the deleted conflict edges to form r components. Let $G'(V, E')$ denote the resulting graph.

   (b) Output G' as the solution.

Fig. 9. An Efficient Algorithm for the M-step Under Model Class II

Theorem 5.2. For the model class consisting of graphs with at most r connected components, the M-step can be carried out in polynomial time. ■

6. Extensions to Multi-valued Problems And Other Variations

In this section, we begin by sketching how our work can be extended to the case where node labels are chosen from a set with three or more values. Suppose the label ℓ_i for node v_i is chosen from the set $\{1, \ldots, m\}$ of m values. We can define the Hamiltonian and the probability mass function by generalizing Equations (2) and (3) as follows.

$$H_M(G) = - \sum_{\{v_i, v_j\} \in E} \delta(\ell_i, \ell_j). \tag{4}$$

$$P_M(G) = \frac{e^{H_M(G)}}{\sum_{G'} e^{H_M(G')}} \tag{5}$$

where $\delta(\ell_i, \ell_j) = +1$ if $\ell_i = \ell_j$ and -1 otherwise. The probability of the graph is still a function of the number of conflicts (N_c) and agreements (N_a) between edges.

With the above definition of $H_M(G)$, the optimization problem associated with the E-step can be modeled by the **multi-way cut** problem [23]. Instead of the two special nodes (s and t) in the binary case, there are m special nodes $s_1, \ldots, s_m$ with the node s_i being connected to each node whose label is ℓ_i. The goal of the multi-way cut problem is to remove a set of edges of minimum total cost so that for every pair s_i and s_j of special nodes, there is no path between s_i and s_j. The multi-way cut problem is **NP**-complete but there exist efficient algorithms that produce provably near-optimal solutions. For example, reference [23] presents an algorithm that produces a cut whose cost is at most twice the optimal cut for all instances of the problem and reference [4] provide a heuristic solution.

Previously, the nodes with missing values ($V - X$) were not connected to any

anchor nodes (i.e. nodes with specific label values). The node's value would be determined by the other nodes values in its connected component. However, if we prefer the node with a missing value to take only one of several values (not any value) we would then connect it those anchor nodes with a small edge weight.

The M-step algorithms of Section 5 can also be used when the number of different node labels is three or more since those algorithms rely only on whether an edge is a conflict edge or an agreement edge.

7. Empirical Results

The EM algorithm has been shown to be useful in a wide variety of settings and we shall simulate several in this section. Our previous theoretical results have proven that the E-step and M-step are optimal. It is then left to empirically validate that the approach works i.e our model classes are useful. Like all applications of EM determining the best modeling assumptions is a key question and we do not claim all our model classes are useful for all problems. Instead we shall use what we believe is the most useful.

We explore several core problems listed below in the context of a sensor network simulator only since it allows us to simulate a limited hardware platform with no floating point units. These results are generalizable to other resource constrained platforms.

– Correcting noisy readings
– Clustering

We present results for our work using the TOSSIM sensor network simulator and MATLAB.

We begin by creating an artificial sensor network on a uniformly spaced 50×50 grid, with a sensor node at each grid point. Larger grids produce similar results, but visually presenting the results for these larger grids is difficult.

We shall use *model class 1* with $\Delta = 8$ meaning that each instance should have at least eight neighbors. Which eight neighbors are chosen is determined by the algorithm and vary from node to node. This differentiates our work from the image processing area where a fixed and repeated neighborhood is used (see Related Work, section 8). for our first application to repair noisy values we use a simple *BOX* example (see Figure 10 for the ideal values), we then randomly removed 15% of all values and then applied our non-parametric EM algorithm to restore them and find the clusters. The **worst** results are shown in Figures 11 and 12. We repeated this experiment twenty times and found that on average only 6% of all missing values (about out 20 out of 375 missing entries) were restored to their **incorrect** values for this simple problem.

We then use our *model class 2* for the problem of finding clusters in the data and repairing missing values. For our *CIRCLES* example (see Figure 13), we randomly removed 15% of all values and then applied our algorithms to restore them. As with all clustering problems, determining k is important. In our context k is set by the model class parameter r which upper bounds k. For values of r to approximately eight the correct results obtained as given by examples before and after the restoration data sets in Figures 14 and 15. We found that on average only 0.4% of all missing values were **not** restored to their correct values after twenty experiments. However, if r is set incorrectly, just as if k is set incorrectly we get less desirable results are shown in Figure 16.

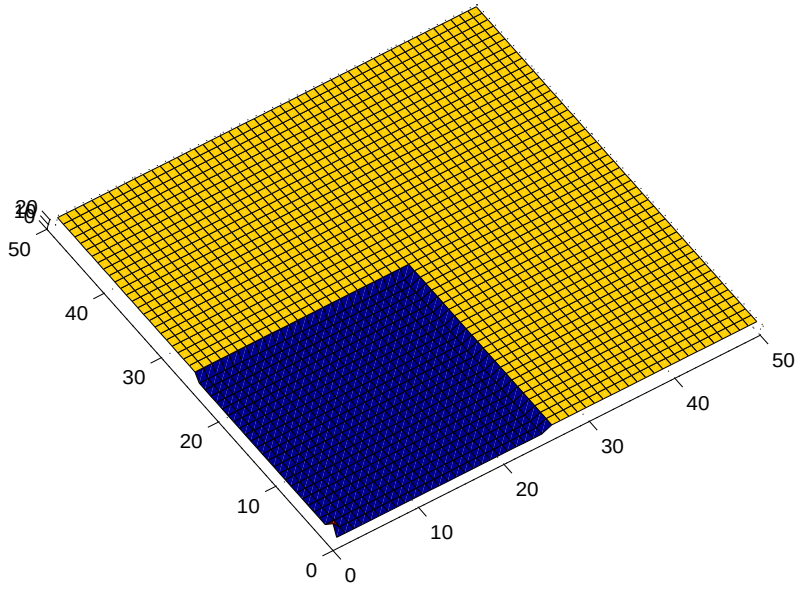

Fig. 10. Ideal Uncorrupted Box Data for a 50×50 uniformly spaced grid.

Finally, we use *model class 1* for the missing value problem shown in Figure 1 with δ set to four. The results are shown in Figure 17.

8. Related Work

Our work is most related to work in computer vision that use Markov Random Fields (MRF). In computer vision MRFs are used to correct noisy (not missing) values by performing the inference $P(f|O)$ where f are the true values and O are the observed values. In reference [4] the authors use st-mincut to perform such an inference for MAP estimation. Our work differs from theirs in several significant ways. Firstly, we use a different energy function that does not use a floating point computation. It turns out that both our objective function and theirs (see equation 3 in [4]) can be effectively addressed using st-mincut. Secondly, their algorithm only performs an inference (E-step) step with no M-step. To the best of our knowledge our work is the first to use st-mincut in the context of an EM algorithm. Finally, as is typically the case for computer vision, a **fixed** model structure is used where each pixel is always connected to its eight neighbors (when arranged in a grid). However, in our work the neighborhood of a pixel dynamically changes depending on which model class is used and changes after each M-step.

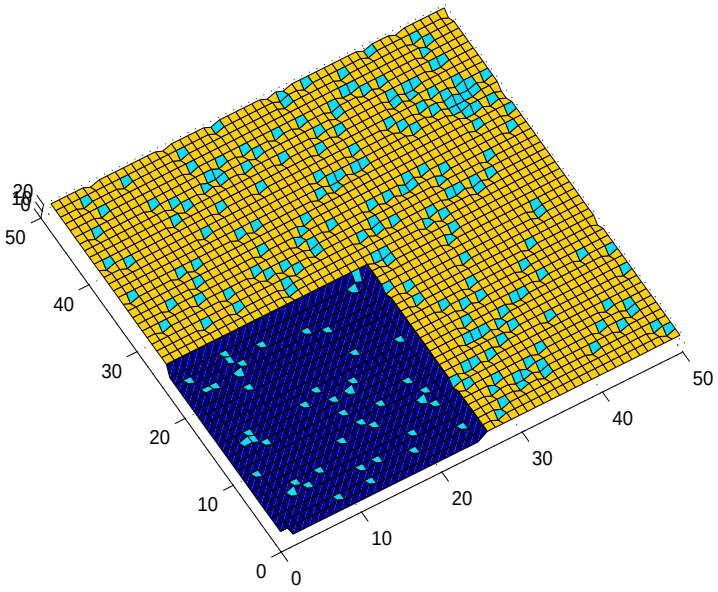

Fig. 11. Box Data with Missing Values for a 50×50 uniformly spaced grid. The odd-colored entries indicates a missing value.

9. Conclusions

The EM algorithm is used extensively in data mining to overcome problems with inherent uncertainty due to missing values or latent variables. However, it cannot be readily applied in its parametric form on many computing platforms that lack floating point hardware, have slow processors or have limited battery life.

We explored non-parametric EM using graphs as model spaces. Model classes correspond to various constraints on the graph topology and we discussed several that we believe have practical benefits. We derived an E-step for all model classes for binary data and discussed an extension for multi-valued data. We wish to fully utilize resources such as time or battery consumption and hence our M-step for model classes I and II **maximized** the complete likelihood.

Finally, we showed our algorithms and model classes performs well on several classical (but simplified) tasks that EM is well known to address.

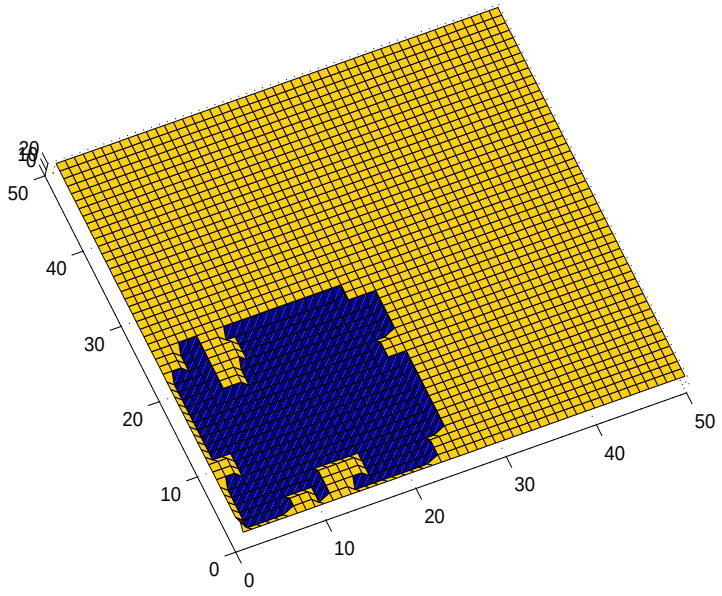

Fig. 12. Recovered Box Data for a 50×50 uniformly spaced grid.

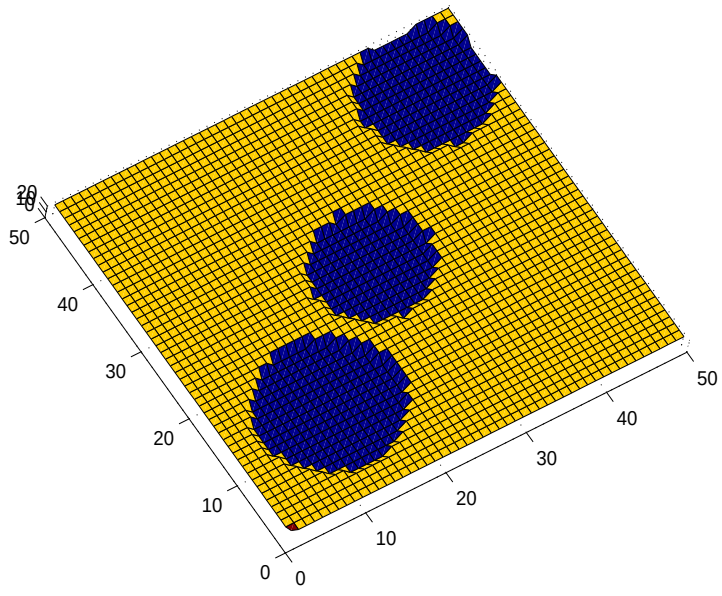

Fig. 13. Original Multiple Circles Data for a 50×50 uniformly spaced grid.

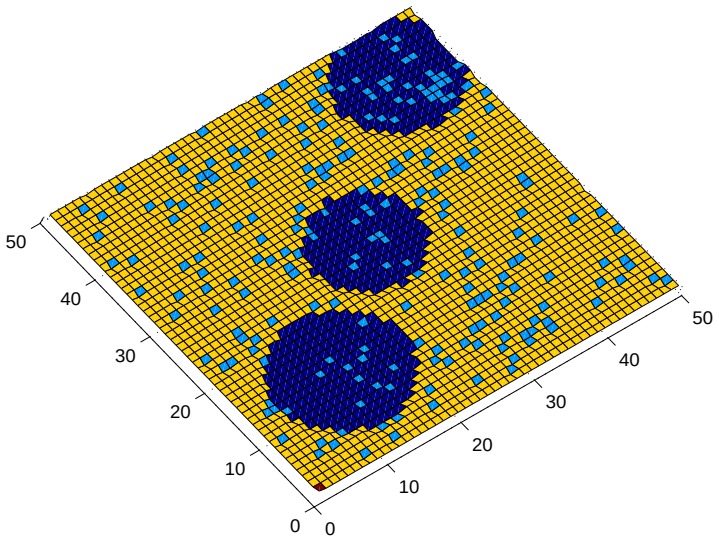

Fig. 14. Multiple Circles Data with Missing Values for a 50×50 uniformly spaced grid. The odd-colored symbols indicate a missing value.

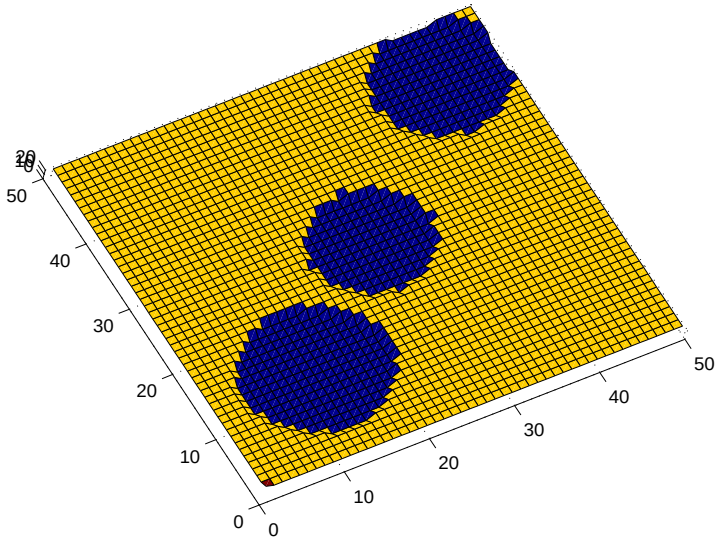

Fig. 15. Recovered Multiple Circles Data for a 50×50 uniformly spaced grid. Clustered and missing values restored for model class II using $r = 6$.

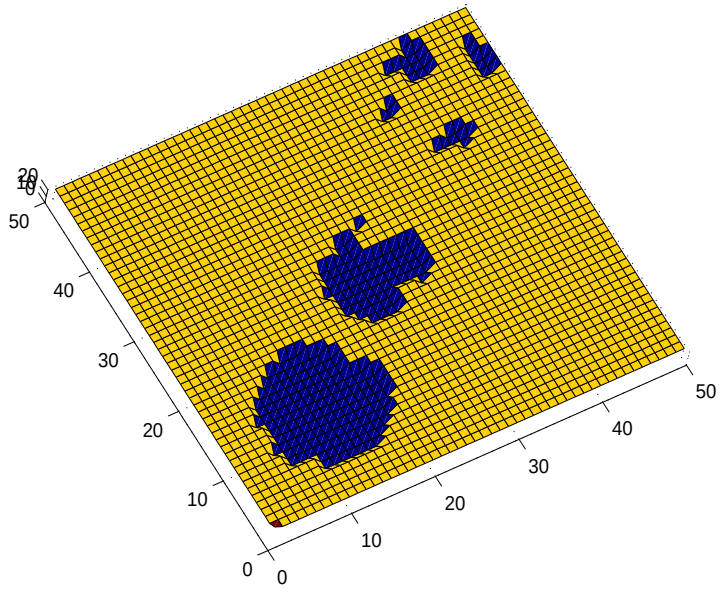

Fig. 16. Recovered Multiple Circles Data for a 50×50 uniformly spaced grid. Clustered and missing values restored for model class II using $r = 20$.

Fig. 17. Recovered Cake Image for a 1200x800 pixel image. Missing values restored for model class I using $\delta = 4$. Compare with Figure 1.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, Vol. 38, 2002.

[2] M. Amini, P. Gallinari, Semi-Supervised Learning with Explicit Misclassification Modeling, *IJCAI'03*.

[3] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, Sonar-Based Mapping with Mobile Robots Using EM, *Proceeding ICML'99*.

[4] Y. Boykov, O. Veksler, R. Zabih, Markov Random Fields with Efficient Approximations, *Proceeding of IEEE Computer Vision and Pattern Recognition*, 1998, , pp. 648-655

[5] R. Caruna, "A Non-Parametric EM-Style Algorithm for Imputing Missing Values", AI-Stats 2001.

[6] B. Cipra, An Introduction to the Ising Model, American Mathematical Monthly, Vol 94 No. 10. 1987.

[7] G. Celeux, G. Govaert, A Classification EM algorithm for Clustering and Two Stochastic Versions, Computational Statistics and Data Analysis, Volume 14, Issue 3, 1992.

[8] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, Cambridge, MA, 2001.

[9] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, Wiley, 2000.

[10] I. N. Davidson and S. S. Ravi, "Distributed Pre-Processing of Data on Networks of Berkeley Motes Using Non-Parametric EM", *Workshop on Data Mining in Sensor Networks*, co-located with SIAM DM'05. (Available from: `www.cs.albany.edu/~davidson/SIAMWS.pdf`)

[11] E. Elnahrawy and B. Nath, "Cleaning and Querying Noisy Sensors", *WSNA'03*.

[12] D. Heckerman, "A Tutorial on Learning with Bayesian Networks", Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[13] P. Klein (Editor), "Combinatorial Optimization: Lecture Notes", Computer Science Department, Brown University, 1990.

[14] D. Koller and A. Pfeffer, Learning probabilities for noisy first-order rules, *IJCAI97*.

[15] M. Meila and D. Heckerman, An Experimental Comparison of Several Clustering and Initialization Methods. *Machine Learning*. 42:9-29, 2001

[16] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[17] A. Moore, Very fast EM-based mixture model clustering using multi-resolution kd-trees, *NIPS99*.

[18] L. Lovasz and M. Plummer, *Matching Theory*, Annals of Discrete Mathematics, Vol 29, 1986.

[19] G. McLachlan. Discriminant Analysis and Statistical Pattern Recognition. John Wiley and Sons, Inc. New York, 1992.

[20] Schumitzky, A., Non-Parametric EM Algorithms for Estimating Prior Distributions, TR 90-2, Department of Mathematics, University of Southern CA.

[21] M. Stoer and F. Wagner, "A Simple Min-Cut Algorithm", *J. ACM*, Vol. 44, No. 4, July 1997.

[22] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P.Yu, Z. Zhou, M. Steinbach, D. Hand and D. Steinberg, Top 10 algorithms in data mining, *Knowledge and Information Systems*, Volume 14, Number 1 / January, 2008, pp. 1-37.

[23] V. V. Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin, 2001.