Filling in the Blanks – KRIMP Minimisation for Missing Data

Jilles Vreeken and Arno Siebes
Department of Computer Science, Universiteit Utrecht
{jillesv,arno}@cs.uu.nl

Abstract

Many data sets are incomplete. For correct analysis of such data, one can either use algorithms that are designed to handle missing data or use imputation. Imputation has the benefit that it allows for any type of data analysis. Obviously, this can only lead to proper conclusions if the provided data completion is both highly accurate and maintains all statistics of the original data.

In this paper, we present three data completion methods that are built on the MDL-based KRIMP *algorithm. Here, we also follow the MDL principle, i.e. the completed database that can be compressed best, is the best completion because it adheres best to the patterns in the data.*

By using local patterns, as opposed to a global model, KRIMP *captures the structure of the data in detail. Experiments show that both in terms of accuracy and expected differences of any marginal, better data reconstructions are provided than the state of the art, Structural EM.*

1 Introduction

Many data sets are incomplete. Whether dealing with surveys, DNA micro arrays or medical data, missing values are commonplace. However, properly dealing with missing values remains an open problem in data analysis. While some specialised algorithms exist that are designed to handle missing data, many are not, and can at most ignore missing values. From statistics, we know [6] this leads to biases in the outcome of the analysis.

There are two ways to properly analyse incomplete data: 1) using specialised algorithms designed to handle missing data or 2) through completing the data by imputation. Of these two, imputation has the practical advantage that one can analyse the completed database using any tool or method desired. Obviously, this does require the imputation to be as accurate as possible. That is, all statistics that one computes from the completed database should be as close as possible to those of the original data. The problem of imputation is thus: complete the database as well as possible.

However, determining what is 'good' cannot just be measured through accuracy: only for 100% correct estimations we know for sure that all statistics are maintained. In this paper we consider 0/1 databases in particular and categorical databases in general. This allows us to properly validate the quality of a completed database in the following manner: we compare the support of a random item set in the original (complete) database with its support in the completed database. The rationale is as follows. Analysing binary data is largely based on counting. If the support of all item sets are correct, all counts will be correct. So, if for random item sets that difference in support is nil, we know that all counts are identical. Consequently, all statistics computed on the completed database will be correct.

To achieve such high quality imputation we use the practical variant of Kolmogorov complexity, MDL (minimum description length), as our guiding principle: the completed database that can be compressed best is the best completion. The driving thought behind this approach is that a completion should comply to the local patterns in the database: not just filling in what globally would lead to the highest accuracy. By taking into account how specific values co-occur locally, not only the global statistics on the data will be correct but also those measured on the local level.

We approximate this best result using KM, which stands for KRIMP Minimisation. It is an iterative approach in which each successive completion has a lower complexity as measured through the compressibility of the data. KM is built on the MDL-based KRIMP algorithm, that provides high quality data descriptions through compression of the data using frequent item sets [10]. The code tables that form the outcome of KRIMP have, amongst other applications, been successfully used in classification [7].

2 The Problem

2.1 Preliminaries

Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of binary (0/1 valued) attributes. That is, the domain D_i of item I_i is $\{0, 1\}$. A transaction (or tuple) over $\mathcal{I}$ is an element of $\prod_{i \in \{1,\ldots,n\}} D_i$. A database $\mathcal{D}$ over $\mathcal{I}$ is a bag of tuples over $\mathcal{I}$. This bag is indexed such that we can talk about the i-th transaction.

An item set J is, as usual, a subset of $\mathcal{I}$, i.e., $J \subseteq \mathcal{I}$. The item set J *occurs* in a transaction $t \in \mathcal{D}$ if $\forall I \in J : \pi_I(t) = 1$. The *support* of item set J in database $\mathcal{D}$ is the number of transactions in $\mathcal{D}$ in which J occurs. That is, $supp_{\mathcal{D}}(J) = |\{t \in \mathcal{D} | \ J \text{ occurs in } t\}|$. An item set is called *frequent* if its support is larger than some user-defined threshold called the *minimal support* or *min-sup*. Given the A Priori property frequent item sets can be mined efficiently.

Note that while we restrict ourself to binary databases in the description of our problem and algorithms, there is a trivial generalisation to categorical databases. In the experiments, we use such categorical databases.

2.2 Missing Data

A database $\mathcal{D}$ has *missing data,* if some of its values are denoted by '?'. The ?-values denote that we do not know what the actual value is, it might be a 0 or a 1. In the traditional market-basket example, this means, e.g., that we do not know whether *Beer* was bought in a given transaction.

The literature, see, e.g., [8,9], distinguishes the following three different types of missing data mechanisms.

MCAR which stands for *missing completely at random.* It means that the fact that a data value is missing does not depend on any of the values in the transaction, including itself.

MAR which stands for *missing at random.* This means that the fact that a data value is missing may depend on one or more of the observed values, it does *not* depend on the "true" value of any of the missing values.

NMAR which stands for *not missing at random.* This means that the fact that a data value is missing may depend on the true value of a missing data value.

NMAR is a very problematic case. Without background knowledge, unbiased analysis of the data is impossible. As the vast majority of the literature, we restrict ourselves to MAR and MCAR only.

2.3 Database Completion

Completing a database simply means that each question mark is replaced by a definite value, i.e., a 1 or a 0. More formally we have the following definition.

Definition 1. *Let $\mathcal{D}$ and $\mathcal{D}_c$ be two databases over $\mathcal{I}$. Moreover, let $\mathcal{D}$ have missing data, whereas $\mathcal{D}_c$ is* complete*, i.e., $\mathcal{D}_c$ has no missing data. Then, $\mathcal{D}_c$ is a* completion *of $\mathcal{D}$ iff*

1. *$\mathcal{D}$ and $\mathcal{D}_c$ both have k transactions, $\mathcal{D} = \{t_1, \ldots, t_k\}$ and $\mathcal{D}_c = \{s_1, \ldots s_k\}$;*
2. *$\forall i \in \{1, \ldots, k\} \ \forall I \in \mathcal{I} : \pi_I(t_i) \in \{0, 1\} \rightarrow \pi_I(t_i) = \pi_I(s_i)$*

An algorithm $\mathcal{A}$ that completes any incomplete database is called a completion algorithm*.*

There are many possible completions of an incomplete database. In fact, if $\mathcal{D}$ has k unknown values, there are 2^k completions. Clearly, not all completions are equally useful. To define quality measures, we assume that we know the true complete database, denoted by $\mathcal{D}^t$. The most obvious quality measure of a completion is accuracy. Clearly, a 100% accurate completion of $\mathcal{D}$ will allow for unbiased estimates on $\mathcal{D}_c$. However, accuracy is a very strict measure. If $\mathcal{D}_c$ is simply a permutation of the rows of $\mathcal{D}^t$, the accuracy can be arbitrarily low. Whereas such a permutation still allows for unbiased estimates. Still, it is the most generally used data completion quality measure [8].

To define a less strict quality measure, recall that the goal of a completion is to allow unbiased statistics. That is, statistics or models computed on $\mathcal{D}_c$ should be as close as possible to their counterparts computed on $\mathcal{D}^t$. Most statistical analysis of categorical data depends crucially on counts and sums. Often subtables are created using selections and projections, and counts and sums on these subtables are computed.

Given that sums are counts on 1's on 0/1 databases , we have the following observation.

Observation 1. *Let $\mathcal{D}, \mathcal{D}^t$, and $\mathcal{D}_c$ be databases over $\mathcal{I}$, such that $\mathcal{D}$ is incomplete, $\mathcal{D}^t$ is the true completion of $\mathcal{D}$ and $\mathcal{D}_c$ is an arbitrary completion of $\mathcal{D}$. If for all item sets $J \subseteq \mathcal{I}$,*

$$supp_{\mathcal{D}_c}(J) = supp_{\mathcal{D}^t}(J)$$

then all counts and sums on project-select subtables on $\mathcal{D}_c$ equal their counterpart on $\mathcal{D}^t$.

With this result in mind, our new quality measure, we can measure how good the support of item sets in a completed database is.

Definition 2. *Let $\mathcal{D}, \mathcal{D}^t$, and $\mathcal{D}_c$ be databases over $\mathcal{I}$, such that $\mathcal{D}$ is incomplete, $\mathcal{D}^t$ is the true completion of $\mathcal{D}$ and $\mathcal{D}_c$ is an arbitrary completion of $\mathcal{D}$. Moreover, let $\epsilon, \delta \in \mathbb{R}$. $\mathcal{D}_c$ is (ϵ, δ)*-correct *if for a random (frequent) item set I*

$$P(|supp_{\mathcal{D}^t}(I) - supp_{\mathcal{D}_c}(I)| > \epsilon) \leq \delta$$

In other words, the support of item sets on (ϵ, δ)-correct completions are almost always close to the correct value. The lower ϵ and δ are, the better the completion is. As an aside, note that (ϵ, δ)-correctness is invariant under permutations; the sum is a commutative operator.

Clearly, if a completion is 100% accurate, it is also $(0, 0)$-correct. If $(0, 0)$-correctness is not attainable, the two measures differ. Due to the invariance of (ϵ, δ)-correctness, it is a more flexible quality measure.

2.4 The Completion Problem

With these quality measures at hand we formalise our completion problem as follows.

The Completion Problem:

Devise a completion algorithm $\mathcal{A}$ that yields an (ϵ, δ)-correct completion for any incomplete database with ϵ and δ as low as possible.

We settle for *as low as possible* because there may not be enough information in the database to derive the $(0, 0)$-correct completion. For example, if $\mathcal{I}$ has only one item, the database has only one transaction and its value is missing. Either $\{1\}$ and $\{0\}$ are possible, and there is no algorithm that will reliably choose correctly. For all practical purposes, though, $(0, 0)$ is well approximable.

3 MDL for Data Completion

In our work, the KRIMP algorithm plays an important role. Recently we introduced this minimum description length (MDL) based algorithm for item set mining [10], although not yet by that name. Due to lack of space we cannot properly introduce the algorithm here, for the convenience of the reader we do introduce its most important notion regarding our current problem.

The MDL principle [5] can be described as follows. Given a set of models $\mathcal{H}$, the best model $H \in \mathcal{H}$ for data D which minimises $L(H) + L(D|H)$, where $L(H)$ is the length, in bits, of the description of H and $L(D|H)$ is the length of the description of the data when encoded with H.

As H, KRIMP uses a set of item sets dubbed a *code table* (CT) to encode and decode the data losslessly. It can be regarded as a table consisting of item sets on the left hand side and the associated optimal code on the right hand side: the more an item set is used, the shorter its code. The item sets are selected such that the total encoded size, $L(CT) + L_{CT}(\mathcal{D})$, is minimised. For more details see [10]. Here in particular we exploit the property of the KRIMP code tables that we calculate the encoded size of individual transactions t, $L_{CT}(t)$. This size can be regarded as a likelihood of the transaction given the data the code table was induced on.

Another way to paraphrase the MDL principle is: the better a model compresses the database, the closer it approximates the underlying data distribution. The key point of both MAR and MCAR is that they do not perturb this underlying distribution. Hence, in the spirit of MDL, one can say: the best completion is the completion that allows for the best compression.

A straightforward implementation of this idea, however, runs the risk of surpressing the natural variation in the data. The more data that is missing, the higher this risk becomes. How susceptible an algorithm is to this risk can be analysed by estimating (ϵ, δ)-correctness. The more susceptible an algorithm is, the worse the (ϵ, δ)-correctness will be.

Next, note that for $\mathcal{D}$ with n missing binary values the search space consists of 2^n possible completions. Clearly, finding the best completion quickly becomes infeasible, even for moderate amounts of missing values. Further, there is no structure that we can exploit to prune this search space. Therefore, we settle for heuristics that approximate the best compressable $\mathcal{D}_c$ by making local decisions.

We introduce three such completion algorithms, based on KRIMP. For the first two algorithms, Simple Completion and KRIMP Completion, we assume that there is enough complete data to allow KRIMP to discover good code tables. Moreover, KRIMP Completion uses randomisation to minimise the risk of variation surpression. The third algorithm, called KRIMP Minimisation, does not rely on the assumption that there is enough complete data.

3.1 Simple Completion

A simple way to impute a missing value is using the maximal likelihood estimator. For KRIMP this reduces to shortest encoded length. Let $t \in \mathcal{D}$ be a transaction with missing values and denote by $C(t)$ the set of all its possible completions. The *Simple Completion* algorithm SC replaces t by that element of $C(t)$ that has the shortest encoded length.

More precisely, let $\mathcal{D} = \mathcal{D}_{comp} \cup \mathcal{D}_{inc}$ such that all transactions in $\mathcal{D}_{comp}$ are complete, while all transactions in $\mathcal{D}_{inc}$ are incomplete. The SC algorithm, Fig. 3(a), first computes a code table CT, by running KRIMP on $\mathcal{D}_{comp}$. Note to ensure any transaction can be encoded we apply a Laplace correction on the code table [7]. Next, each incomplete transaction is replaced by the completed transaction with the shortest encoding.

3.2 KRIMP Completion

By always choosing the most likely candidate, the SC algorithm may have a detrimental effect on the support of some item sets. Suppose, e.g., that the encoded length of $(1, 1)$ is slightly shorter than $(1, 0)$. Then each occurrence of $(1, ?)$ will be replaced by $(1, 1)$ by SC. This leads to an overestimate of the support of $(1, 1)$ and an underestimate of the support of $(1, 0)$. The KRIMP *Completion* algorithm KC, see Fig. 3(b), remedies this by choosing an element of $C(t)$ with a chance proportional to its encoded length. Again KRIMP is first run on $\mathcal{D}_{comp}$. The resulting code table CT defines a probability distribution $P_{CT}(t)$ on $C(t)$ is $P_{CT}(t)(s) = \frac{2^{-L_{CT}(s)}}{\sum_{u \in C(t)} 2^{-L_{CT}(u)}}$. The completion of t is chosen from $C(t)$ according to this distribution. The function CHOICE$(C(t), CT)$ makes this random choice.

```
SC($\mathcal{D}$)
1  $CT$ := KRIMP($\mathcal{D}_{comp}$)
2  foreach $t \in \mathcal{D}_{inc}$
       $t := \text{argmin}_{s \in C(t)} L_{CT}(s)$
3  return $\mathcal{D}_{comp} \cup \mathcal{D}_{inc}$
```

(a) The SC Algorithm

```
KC($\mathcal{D}$)
1  $CT$ := KRIMP($\mathcal{D}_{comp}$)
2  foreach $t \in \mathcal{D}_{inc}$
       $t :=$ CHOICE($C(t), CT$)
3  return $\mathcal{D}_{comp} \cup \mathcal{D}_{inc}$
```

(b) The KC Algorithm

```
KM($\mathcal{D}$)
1  $\mathcal{D}_c$ := random completion of $\mathcal{D}$
2  while not converged
       $CT$ := KRIMP($\mathcal{D}_c$)
       $\mathcal{D}_c$ := KC($\mathcal{D}$)
3  return $\mathcal{D}_c$
```

(c) The KM Algorithm

Figure 1. Pseudo-code for SC (a), KC (b) and KM (c)

3.3 KRIMP Minimisation

For KC to work, we need enough complete data for KRIMP to compute a good code table. If there is not enough complete data, the result of KC may be arbitrarily bad. To handle such a lack of sufficient complete data, we take an EM-like [3] approach.

The KRIMP *Minimisation* algorithm KM starts with a random completion of the incomplete database $\mathcal{D}$. Then it iterates through a number of KRIMP and KC steps. In the KRIMP step it compresses the current *complete* database. In the KC step it completes the *incomplete* database $\mathcal{D}$ using the code table computed in the KRIMP step. This is continued as long as the total encoded length of the completed database shrinks. The algorithm returns the final completed database. It has the shortest encoded length of the considered completions, hence the name of the algorithm.

Note that KM will always terminate. The total encoded size shrinks with every step. Since the encoded size of any finite database is finite, KM can only execute a finite number of iterations.

4 Related Work

One of the first known examples of missing data estimation, hot deck imputation [1], was employed by the US census bureau in the fifties. It replaces missing records by random draws from complete records from the same local area. As such, it may be regarded as a crude form of k nearest-neighbour imputation [12] which also requires a distance function on the data, unlike our methods.

Regression, mean substitution and mean-mode [6] imputation have a greedy nature that harms the variance in the completed data [1]. Using some randomness circumvents this, which is why Expectation Maximisation (EM) [3] is the current state of the art. Imputation through EM is the process of maximising the likelihood of the data given a distribution. Iteratively, it adapts the model to the data and re-imputes it. EM has been shown to provide very accurate probability estimations. KRIMP Minimisation follows a similar iterative approach to optimise the compressed size of the database.

Integrating a structure learner into the EM process leads to even better results. Structural EM (SEM) [4] learns Bayes nets during the modeling phases. SEM has been shown to provide high quality probability estimates, and very good approximations of the original Bayes nets. However, it is computationally expensive and thereby only feasible for moderately sized datasets. Here, we do not learn a global model on the data but consider it in closer detail by using local patterns.

Table 1. Statistics of the data sets used.

				baseline	*%*
Dataset	$\|t\|$	$\|\mathcal{I}\|$	$\|\mathcal{D}\|$	*accuracy*	*min–sup*
alarm	37	105	5000	79.9%	50.0
chess (krk)	7	58	28056	23.0%	0.7
led7	8	24	3200	61.2%	0.03
let.Recog	17	102	20000	57.5%	1.2
mushroom	23	119	8124	57.6%	1.2
penDigits	17	86	10992	35.7%	0.9
wine	14	68	178	43.5%	0.6

5 Experiments

In this section we evaluate the performance of our proposed methods. However, due to space constraints we cannot provide the full empirical analysis, but focus on the hardest problem: incomplete training data. For the full empirical analysis of our methods, as well as providing more details on the theory, we kindly refer the reader to [11].

Unless indicated otherwise, all results in this section are averaged over 10 independent runs. Again, unless indicated otherwise, here we consider the case of on average 1 missing value per transaction. The (ϵ, δ)–correctness is calculated over the closed frequent item set collection as mined on the complete test data. Missing data was created by either removing items fully randomly (MCAR), or removing items randomly dependent on a present item (MAR).

Table 2. Imputation quality measurements using SC and KM.

			MCAR				*MAR*			
			SC		KM		SC		KM	
Dataset	*% miss*	ϵ	δ	*accuracy*	δ	*accuracy*	δ	*accuracy*	δ	*accuracy*
alarm	2.7	0.7	16.4	84.0	3.0	82.5	17.3	85.7	5.4	83.6
chess	14.3	0.1	15.8	35.4	3.6	33.7	18.8	32.7	4.2	28.2
led7	12.5	1.0	32.7	72.7	1.2	79.2	17.2	82.8	1.2	81.1
let.Recog	5.9	0.8	16.4	64.2	4.8	61.9	14.2	67.1	5.9	65.3
mushroom	4.3	0.5	4.4	76.3	4.1	74.2	4.1	74.5	0.3	70.9
penDigits	5.9	0.1	8.8	66.6	3.8	67.2	11.8	64.6	5.3	65.7
wine	7.1	1.1	5.3	55.2	4.9	54.6	6.9	50.3	5.1	48.8

5.1 Datasets

We use a range of data sets to validate our methods. From the widely used UCI repository [2] we took seven databases. Further, for fair comparison to the Bayesian Structural EM method, we generated data from the well-known *alarm* network. This data was generated using the *GenInstance* program, made available by Nir Friedman in the LibB Bayes Network tool library [1].

The details for these data sets are depicted in Table 1. Apart from their base statistics, we provide the baseline imputation accuracy on MCAR data as achieved by choosing the most frequent of the possible values.

We use the closed frequent pattern set as candidates for KRIMP. Being generated from a Bayes net, the *alarm* dataset contains no local structure. The absence hereof leads to a gigantic explosion in the number of patterns; hence we have to use a high min–sup for this database.

5.2 Insufficient Complete Data

We consider the problem for when no (sufficient) complete training data is available, so, we employ SC and KM. The results of these experiments is presented in Table 2. First we can remark that the imputation accuracies are generally higher than with complete training data [11]. However, through (ϵ, δ)–correctness we can see that no magic is going on, as for all datasets these scores actually did decrease; the incomplete training data hinders both methods in grasping the true data distribution.

Also, we see that, as expected, the greedy nature of SC leads to a decrease in the data quality. For all datasets the accuracy provided by KM are only slightly lower than SC. This small loss in accuracy is compensated for by a strong gain in (ϵ, δ)–correctness. For KM, these scores are up to an order better than SC and often approximate the scores attained on the complete training data.

Further investigations regarding the data reconstruction ability of KM were done by looking into the compressed sizes of the data; To compress the data with missing values, KRIMP typically requires 30% more bits than it does to encode the original data. However, through iterative imputation, KM is able to approximate the KRIMP complexity of the original data within a single percent. As noise is canceled, the KM-imputed data has slightly lower complexity than the unseen original.

These experiments also showed that KM rapidly converges to this approximate original complexity: only three iterations were typically required, two more for *mushroom*. Further, we noticed that while KM is nondeterministic in nature, the resulting accuracies, correctness and data complexities are all virtually equal.

5.3 Comparing to SEM

Now that we have verified that KM provides good data completions, we will compare it to the state of the art in imputation: Bayes Structural EM (SEM) [4]. Structural EM incorporates the learning of a Bayes net on within the EM [3] process. It iteratively learns and re-imputes the data, until the process converges.

In order to learn Bayes Nets on incomplete training data, we used Friedman's implementation of SEM. It can be initialised in various ways. As we did not find significant differences in performance, we used random trees. For inference of the missing values we used the Bayes Network Toolbox for Matlab (BNT) on the found networks. As both learning the Bayes nets and inference are computationally expensive, we do not consider all datasets here.

The results of this experiment are presented in Table 3. From it, we first notice that KM attains higher imputation accuracies than SEM for three out of the five datasets. However, the general quality of the KM imputed databases, as measured through the (ϵ, δ)–correctness scores, is quite dramatically better than the Bayes net driven SEM approach. Even for the *alarm* dataset (with relatively few missing values), SEM is unable to score a win. Although this dataset contains no local structure, and we were forced to use high min–sup values for KRIMP, the resulting code tables still

[1]http://www.cs.huji.ac.il/labs/compbio/LibB

Table 3. Imputation quality measurements using KM and Structural EM (SEM).

			MCAR				*MAR*			
			KM		*SEM*		KM		*SEM*	
Dataset	*% miss*	ϵ	δ	*accuracy*	δ	*accuracy*	δ	*accuracy*	δ	*accuracy*
alarm	2.7	0.5	6.7	82.5	15.2	80.9	8.0	83.6	37.8	82.5
chess	14.3	0.1	4.1	33.7	24.5	23.5	4.2	28.2	33.2	22.7
led7	12.5	0.5	0.9	79.2	2.6	76.1	0.8	81.1	2.3	79.1
wine	7.1	1.6	4.9	54.6	7.9	46.1	1.1	48.8	12.8	27.5

allow for better reconstruction of the original data than with the SEM induced global Bayes Net model.

Similar to the previous experiments, no strong trend presents itself when we compare between MCAR and MAR. For KM accuracy is harmed slightly on average, but its (ϵ, δ)-correctness remains stable or even improves. For SEM, however, we do see that for both *alarm* and *chess* the data quality does suffer significantly.

6 Discussion

The experimental results of our methods show that compression is a very viable approach to the data completion problem. All three our parameter-free data completion algorithms show that approximating the best compressible completed database leads to high quality imputation.

Provided with undamaged training data, SC provides highly accurate estimates. The method was shown to be robust: even up to 24% missing values, both accuracy and (ϵ, δ)–correctness of the completed data are very high.

For the realistic case of only insufficient complete training data being available, KRIMP Minimisation is the right choice for a data completion algorithm. It finds good approximations of the best compressible completed database, and is shown to provide both provide high accuracy and very good (ϵ, δ)–scores. Further, the complexity of the original data is approximated within a single percent.

Compared to the state of the art in missing value estimation, Structural EM, the completions that KM offers provide both higher accuracy and adhere better to all count statistics of the original data.

Here we only consider the code tables from the KRIMP compressor. As the proposed methods operate straightforwardly, it is possible to employ other compression schemes instead, for instance to better suit other data types.

7 Conclusions

In this paper we considered the problem of high quality imputation of missing data. To test this objectively we propose (ϵ, δ)–correctness to measure the difference between two databases in terms of count statistics.

We presented three KRIMP–based methods for imputation of incomplete datasets. All follow the MDL–principle: the completed database that can be compressed best is the best completed database. This, because then the imputations adhere to the local patterns that are present in the database, instead of only keeping its global statistics correct. The experiments show that the local approach is indeed superior to the global approach, both in terms of accuracy and quality of the completed databases.

References

[1] P. Allison. *Missing Data – Quantitative Applications in the Social Science*. Sage Publishing, 2001.

[2] F. Coenen. The LUCS-KDD discretised/normalised ARM and CARM data library: http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/. 2003.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[4] N. Friedman. Learning bayesian networks in the presence of missing values and hidden variables. In *Proceedings of ICML*, pages 125–133, 1997.

[5] P. D. Grünwald. Minimum description length tutorial. In P. Grünwald and I. Myung, editors, *Advances in Minimum Description Length*. MIT Press, 2005.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[7] M.van Leeuwen, J. Vreeken, and A. Siebes. Compression picks the item sets that matter. In *Proceedings of ECML PKDD*, pages 585–592, 2006.

[8] R. Little and D. Rubin. *Statistical Analysis with Missing Data (2nd Edition)*. John Wiley and Sons, 2002.

[9] J. Schafer. Analysis of incomplete multivariate data. *Monographs on Statistics and Applied Probability*, 72, 1997.

[10] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proceedings of the SIAM Conference on Data Mining*, pages 393–404, 2006.

[11] J. Vreeken and A. Siebes. Krimp minimisation for missing data estimation. Technical Report UU-CS-2008-034, Universiteit Utrecht, 2008.

[12] I. Wasito and B. Mirkin. Nearest neighbour approach in the least-squares data imputation algorithms. *Journal of Information Sciences*, 167:1–25, 2005.