Clustering with Missing Values: No Imputation Required

Kiri Wagstaff

Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr., Pasadena, CA 91109 `kiri.wagstaff@jpl.nasa.gov`

Summary. Clustering algorithms can identify groups in large data sets, such as star catalogs and hyperspectral images. In general, clustering methods cannot analyze items that have missing data values. Common solutions either fill in the missing values (imputation) or ignore the missing data (marginalization). Imputed values are treated as just as reliable as the truly observed data, but they are only as good as the assumptions used to create them. In contrast, we present a method for encoding partially observed features as a set of supplemental soft constraints and introduce the KSC algorithm, which incorporates constraints into the clustering process. In experiments on artificial data and data from the Sloan Digital Sky Survey, we show that soft constraints are an effective way to enable clustering with missing values.

1 Introduction

Clustering is a powerful analysis tool that divides a set of items into a number of distinct groups based on a problem-independent criterion, such as maximum likelihood (the EM algorithm) or minimum variance (the k-means algorithm). In astronomy, clustering has been used to organize star catalogs such as POSS-II (Yoo et al., 1996) and classify observations such as IRAS spectra (Goebel et al., 1989). Notably, the Autoclass algorithm identified a new subclass of stars based on the clustering results (Goebel et al., 1989). These methods can also provide data compression or summarization by quickly identifying the most representative items in a data set.

One challenge in astronomical data analysis is *data fusion*: how to combine information about the same objects from various sources, such as a visible-wavelength catalog and an infra-red catalog. A critical problem that arises is that items may have missing values. Ideally, each object in the sky should appear in both catalogs. However, it is more likely that some objects will not. The two instruments may not have covered precisely the same regions of the sky, or some objects may not emit at the wavelengths used by one of the catalogs. Missing values also occur due to observing conditions, instrument sensitivity limitations, and other real-world considerations.

Clustering algorithms generally have no internal way to handle missing values. Instead, a common solution is to fill in the missing values in a pre-processing step. However, the filled-in values are inherently less reliable than the observed data. We propose a new approach to clustering that divides the data features into *observed features*, which are known for all objects, and *constraining features*, which contain missing values. We generate a set of constraints based on the known values for the constraining features. A modified clustering algorithm, KSC (for "K-means with Soft Constraints"), combines this set of constraints with the regular observed features. In this paper, we discuss our formulation of the missing data problem, present the KSC algorithm, and evaluate it on artificial data as well as data from the Sloan Digital Sky Survey. We find that KSC can significantly outperform data imputation methods, without producing possibly misleading "fill" values in the data.

2 Background and Related Work

Green et al. (2001) (among others) identified two alternatives to handling missing values: *data imputation*, where values are estimated to fill in missing values, and *marginalization*, where missing values are ignored. However, imputed data cannot and should not be considered as reliable as the actually observed data. Troyanskaya et al. (2001) stated this clearly when evaluating different imputation methods for biological data: "However, it is important to exercise caution when drawing critical biological conclusions from data that is partially imputed. [...] [E]stimated data should be flagged where possible [...] to avoid drawing unwarranted conclusions."

Despite this warning, data imputation remains common, with no mechanism for indicating that the imputed values are less reliable. One approach is to replace all missing values with the observed mean for that feature (also known as the "row average" method in DNA microarray analysis). Another method is to model the observed values and select one according to the true distribution (if it is known). A more sophisticated approach is to infer the value of the missing feature based on that item's observed features and its similarity to other (known) items in the data set (Troyanskaya et al., 2001). Ghahramani and Jordan (1994) presented a modified EM algorithm that can process data with missing values. This method simultaneously estimates the maximum likelihood model parameters, data cluster assignments, and values for the missing features. Each of these methods suffers from an inability to discount imputed values due to their lack of full reliability.

We therefore believe that marginalization, which does not create any new data values, is a better solution. Most previous work in marginalization has focused on supervised methods such as neural networks (Tresp et al., 1995) or Hidden Markov Models (Vizinho et al., 1999). In contrast, our approach handles missing values even we have no labeled training data. In previous work, we developed a variant of k-means that produces output guaranteed

to satisfy a set of hard constraints (Wagstaff et al., 2001). Hard constraints dictate that certain pairs of items must or must not be grouped together. Building on this work, we present an algorithm that can incorporate soft constraints, which indicate how strongly a pair of items should or should not be grouped together. In the next section, we will show how this algorithm can achieve good performance when clustering data with missing values.

3 Clustering with Soft Constraints

Constraints can effectively enable a clustering algorithm to conform to background knowledge (Wagstaff et al., 2001; Klein et al., 2002). Previous work has focused largely on the use of hard constraints that must be satisfied by the algorithm. However, in the presence of uncertain or approximate information, and especially for real-world problems, soft constraints are more appropriate.

3.1 Soft Constraints

In this work, we divide the feature set into F_o, the set of observed features, and F_m, the set of features with missing values. We also refer to F_m as the set of *constraining* features, because we use them to constrain the results of the clustering algorithm; they represent a source of additional information.

Following Wagstaff (2002), we represent a soft constraint between items d_i and d_j as a triple: $\langle d_i, d_j, s \rangle$. The strength, s, is proportional to distance in F_m. We create a constraint $\langle d_i, d_j, s \rangle$ between each pair of items d_i, d_j with values for F_m, where

$$s = -\sqrt{\sum_{f \in F_m} (d_i.f - d_j.f)^2} \qquad (1)$$

We do not create constraints for items that have missing values. The value for s is negative because this value indicates the degree to which d_i and d_j should be separated. Next, we present an algorithm that can accommodate these constraints while clustering.

3.2 K-means Clustering with Soft Constraints

We have chosen k-means (MacQueen, 1967), one of the most common clustering algorithms in use, as our prototype for the development of a soft constrained clustering algorithm. The key idea is to cluster over F_o, with constraints based on F_m.

The k-means algorithm iteratively searches for a good division of n objects into k clusters. It seeks to minimize the total variance V of a partition, i.e., the sum of the (squared) distances from each item d to its assigned cluster C:

Table 1. KSC algorithm

KSC(k, D, SC, w)

1. Let $C_1 \ldots C_k$ be the initial cluster centers.
2. For each instance d in D, assign it to the cluster C such that:

$$C := \underset{C_i}{\operatorname{argmin}} \left((1-w)\frac{dist(d, C_i)^2}{V_{max}} + w\frac{CV_d}{CV_{max}} \right) \qquad (2)$$

   where CV_d is the sum of (squared) violated constraints in SC that involve d.
3. Update each cluster center C_i by averaging all of the points $d_j \in C_i$.
4. Iterate between (2) and (3) until convergence.
5. Return the partition $\{C_1 \ldots C_k\}$.

$$V = \sum_{d \in D} dist(d, C)^2$$

Distance from an item to a cluster is computed as the distance from the item to the center of the cluster. When selecting the best host cluster for a given item d, the only component in this sum that changes is $dist(d, C)^2$, so k-means can minimize variance by assigning d to the closest available cluster:

$$C = \underset{C_i}{\operatorname{argmin}} \ dist(d, C_i)^2$$

When clustering with soft constraints, we modify the objective function to penalize for violated constraints. The KSC algorithm takes in the specified number of clusters k, the data set D (with F_o only), a (possibly empty) set of constraints SC, and a weighting factor w that indicates the relative importance of the constraints versus variance (see Table 1).

KSC uses a modified objective function f that combines normalized variance and constraint violation values. The normalization enables a straightforward specification of the relative importance of each source, via a weighting factor $w \in [0, 1]$.

$$f = (1-w)\frac{V}{V_{max}} + w\frac{CV}{CV_{max}} \qquad (3)$$

Note that w is an overall weight while s is an individual statement about the relationship between two items. The quantity CV is sum of the squared strengths of violated constraints in SC. It is normalized by CV_{max}, the sum of all squared constraint strengths, whether they are violated or not. A negative constraint, which indicates that d_i and d_j should be in different clusters, is violated if they are placed into the same cluster. We also normalize the variance of the partition by dividing by V_{max}, the largest possible variance given D. This is the variance obtained by assigning all items to a single cluster.

In deciding where to place item d, the only constraint violations that may change are ones that involve d. Therefore, KSC only considers constraints in which d participates and assigns items to clusters as shown in Equation 2.
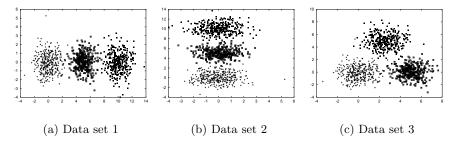
(a) Data set 1   (b) Data set 2   (c) Data set 3

Fig. 1. Three artificial data sets used to compare missing data methods.

4 Experimental Results

We have conducted experiments on artificial data as well as observational data from the Sloan Digital Sky Survey. As we will show, KSC in combination with constraints can often outperform typical data imputation methods. We will compare our approach to three common methods for handling missing data.

The first method, NOMISSING, does not impute data. Instead, it discards all of the features in F_m, relying only on the features in F_o. In doing so, it may discard useful information for the items that do possess values for F_m. Since our approach also bases its clustering only on features in F_o, any difference in performance will be due to the effect of including the soft constraints.

The second approach is MEANVALUE, which replaces each missing value with the mean of the observed values for that feature. The third approach is PROBDIST, which replaces missing values based on the observed distribution of values. For each item d with a missing value for feature f, we sample randomly from the observed values for f and select one to replace the missing value $d.f$.

4.1 Partitioning Artificial Data

We created three artificial data sets (see Figure 1). In each case, the data set contains 1000 items in three classes that form Gaussian distributions. Data set 1 consists of three classes that are separable according to feature 1 (along the x-axis). Data set 2 consists of the same data mirrored so that the classes are separable according to feature 2. Data set 3 is data set 1 except with the rightmost class moved up and between the other two classes. In the final case, the three classes are not separable with either feature in isolation.

We have tested each method by generating variants of these data sets that contain increasing fractions of randomly missing values. For consistency, we only remove values in feature 2 (the y-axis). Figure 2 illustrates the problems associated with both imputation methods by showing the resulting data sets when 50% of the feature 2 values for items in data set 2 are removed, then imputed. MEANVALUE positions all of the imputed data in the middle cluster, while PROBDIST thoroughly mixes the data between clusters.
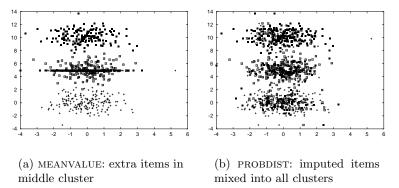
(a) MEANVALUE: extra items in middle cluster

(b) PROBDIST: imputed items mixed into all clusters

Fig. 2. Imputed data for data set 2 when 50% of feature 2's values are missing.

To create the soft constraints for KSC, we examine each pair of items d_i, d_j. If both have values for feature 2, we create a soft constraint between them proportional to their distance in feature 2 (cf. Equation 1):

$$s = -|d_i.f_2 - d_j.f_2| \tag{4}$$

The constraint is negative because it indicates how likely the two items are to be in different clusters. We then cluster using only feature 1, similar to NOMISSING, but with the additional constraint information.

We also need to select a value for w, the weight that is given to the constraint information. For data set 1, we know that feature 2 is not very useful for distinguishing the clusters, so we set $w = 0.01$. For data set 2, feature 2 is critical, so we set $w = 0.99$. For data set 3, where both features are important, we use the "equal weight" default value of $w = 0.5$. If the importance of F_m is unknown, it can be estimated by clustering with several different values for w on a small labeled subset of the data.

To evaluate performance, we compare the resulting partition (P_1) to the true partition (P_2) via the adjusted Rand index (Hubert and Arabie, 1985), averaged over ten trials. Let n_{ij} be the number of items that appear in cluster i in P_1 and in cluster j in P_2. Then R is the total agreement between P_1 and P_2, and we calculate

$$Rand(P_1, P_2) = \frac{R - E[R]}{M[R] - E[R]}, \ R = \sum_{ij} \binom{n_{ij}}{2}$$

where $E[R] = \left[\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2} \right] / \binom{n}{2}$ is the expected value of R and $M[R] = \frac{1}{2} \left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2} \right]$ is the maximum possible value for R.

Because data set 1 is readily separable without any information from feature 2, we found that performance was identical for all methods at all fractions
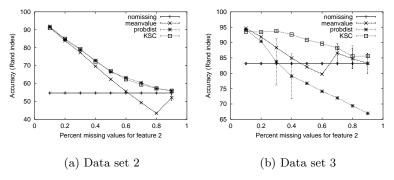
(a) Data set 2   (b) Data set 3

Fig. 3. Accuracy for all four methods on two artificial data sets ($k = 3$, 10 trials); bars indicate $+/-$ one standard deviation.

of missing values, at 98.8%. However, we observe dramatic differences for the other two data sets (see Figure 3). Performance for NOMISSING is independent of the fraction of missing values, because it only uses feature 1 to cluster the data. For data set 2, PROBDIST and KSC perform almost identically, though they make different kinds of errors. The output of PROBDIST assigns items with missing values arbitrarily to one of the three clusters. KSC's output comes to resemble that of NOMISSING when less constraint information is available (i.e., more values are missing). MEANVALUE outperforms the other methods for missing fractions $\geq 50\%$, largely by placing the majority of the items all into the middle cluster. Although items from the top and bottom clusters are not correctly clustered, all of the items in the middle cluster are. This could be an artifact of using three clusters.

We find that KSC and MEANVALUE outperform PROBDIST for data set 3. In addition, KSC is much more robust to an increasing fraction of missing values. It is consistently the best performer for missing fractions greater than 10%. Overall, KSC performs as well or better than discarding or imputing data, without generating any potentially misleading data values. The exception to this trend is for data where the clusters are not separable in the observed features, when little information is available as constraints.

4.2 Separating Stars from Galaxies

In addition to our experiments with artificial data, we have also used KSC to analyze a portion of the SDSS star catalog. Our data set contains 977 objects: 354 stars and 623 galaxies. The goal is to produce a partition that separates stars from galaxies. Each object is represented by 5 features: brightness (point spread function flux), size (Petrosian radius, in arcsec), texture (small-scale roughness of object), and two shape features, all observed at 365 nm. To obtain this data set, we used the SDSS recommendations to obtain a "clean" sample and excluded faint stars (flux of > 20 magnitude) and very bright

galaxies (flux of ≤ 20 magnitude). This restriction on the data allowed us to get a sample that is fairly separable based on the features we used.

The SDSS data set contains missing values in the shape features, affecting 49 of the objects in our data set. However, the missing values only occur for stars. Ghahramani and Jordan (1994) noted that no general data imputation method can handle missing data that is confined to a single class, which is another reason to favor non-imputation methods. The fully observed features (F_o) are brightness, size, and texture. We created imputed versions of the data set for MEANVALUE and PROBDIST as described in the previous section. We created the set of soft constraints by generating, for each pair d_i, d_j that did have values for both features in F_m, a soft constraint according to Equation 1. The resulting set contained $873,920$ constraints. Without any specific guidance as to the importance of F_m, we used a default value of $w = 0.5$.

However, in examining the output of each method we discovered that better performance could be obtained when clustering the data into *three* clusters, as shown in the following table. Three clusters allow finer divisions of the data, though splitting stars or galaxies across clusters is still penalized. This result suggests that there is a distinct star type that is easily confused with galaxies. The third cluster permitted these ambiguous items to be separated from the true star and galaxy clusters.

Method	NOMISSING	MEANVALUE	PROBDIST	KSC, $w = 0.5$
$k = 2$	80.6	70.6	70.6	68.3
$k = 3$	88.0	82.6	81.5	84.2

In both cases, NOMISSING achieved the best performance, suggesting that the shape features may not be useful in distinguishing stars from galaxies. To test this theory, we ran KSC with w values ranging from 0.1 to 0.9 (see Figure 4). We find that shape information *can* be useful, if properly weighted. For $w = 0.3, 0.4$, KSC outperforms NOMISSING, attaining a peak value of 90.4%. Otherwise, NOMISSING is the best method. There is no general rule for selecting a good w value, but existing knowledge about the domain can point to a value.

The large standard deviation for KSC is due to the occasional solution that does not satisfy very many constraints. For example, when $w = 0.4$, four of 50 trials converged to a solution that satisfied only 23% of the constraints (Rand 62%). The remaining trials satisfied about 46% of the constraints and achieved 90% performance. Thus, the constraint information, when satisfied, is very useful. MEANVALUE and PROBDIST also have standard deviations of about 5.0 (not shown in Figure 4 for clarity).

5 Conclusions and Future Work

In this paper, we presented a new approach to the missing value problem for clustering algorithms. We have discussed, and demonstrated, the difficulties
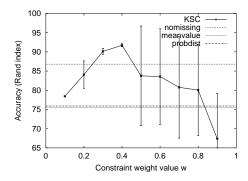
Fig. 4. Accuracy when clustering SDSS data with different weights for the constraints ($k = 3$, 50 trials); bars indicate $+/-$ one standard deviation for KSC.

of data imputation methods that process imputed values as if they were as reliable as the actual observations. We presented a new approach that divides the data features into F_o, the observed features, and F_m, the features with missing values. We generate a set of soft constraints based on distances calculated on known values for F_m. A new algorithm, KSC, can apply these soft constraints. In experiments with artificial data and the SDSS star/galaxy catalog, we have shown that KSC can perform as well as or better than data imputation methods. In some cases, knowledge of the relative importance of the missing features is necessary.

In future work, we plan to compare KSC directly to the EM-based algorithm of Ghahramani and Jordan (1994). Although their goals were different (recovering missing data values vs. determining the correct partition), a comparison would provide further understanding of the applicability of KSC.

Another consideration is the complexity of the KSC approach. The number of constraints is $O(n^2)$, where n is the number of items that possess values for features in F_m. Klein et al. (2002) have suggested a technique for propagating hard constraints through the feature space, obtaining equivalent results with fewer explicit constraints. Although their work was restricted to hard constraints, we would like to explore options for extending their method to work with soft constraints as well.

Acknowledgments

Institutions are The University of Chicago, Fermilab, the Institute for Advanced Study, the Japan Participation Group, The Johns Hopkins University, Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy (MPIA), the Max-Planck-Institute for Astrophysics (MPA), New Mexico State University, University of Pittsburgh, Princeton University, the United States Naval Observatory, and the University of Washington.

References

Ghahramani, Z. and Jordan, M. I. (1994). Learning from incomplete data. Tech. Rep., Massachusetts Inst. of Technology Artificial Intelligence Lab.

Goebel, J., Volk, K., Walker, H., Gerbault, F., Cheeseman, P., Self, M., Stutz, J. and Taylor, W. (1989). A Bayesian classification of the IRAS LRS Atlas. *Astronomy and Astrophysics* **222** L5–L8.

Green, P. D., Barker, J., Cooke, M. P. and Josifovski, L. (2001). Handling missing and unreliable information in speech recognition. In *Proc. of AISTATS 2001*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification* **2** 193–218.

Klein, D., Kamvar, S. D. and Manning, C. D. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. of the 19th Intl. Conf. on Machine Learning*, 307–313. Morgan Kaufmann.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability*, vol. 1, 281–297. University of California Press.

Tresp, V., Neunier, R. and Ahmad, S. (1995). Efficient methods for dealing with missing data in supervised learning. In *Advances in Neural Info Proc. Sys. 7*.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. and Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics* **17** 520–525.

Vizinho, A., Green, P., Cooke, M. and Josifovski, L. (1999). Missing data theory, spectral subtraction and signal-to-noise estimation for robust ASR: An integrated study. In *Proc. of Eurospeech '99*, 2407–2410.

Wagstaff, K. (2002). *Intelligent Clustering with Instance-Level Constraints*. Ph.D. dissertation, Cornell University.

Wagstaff, K., Cardie, C., Rogers, S. and Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *Proc. of the 18th Intl. Conf. on Machine Learning*, 577–584. Morgan Kaufmann.

Yoo, J., Gray, A., Roden, J., Fayyad, U., de Carvalho, R. and Djorgovski, S. (1996). Analysis of Digital POSS-II Catalogs using hierarchical unsupervised learning algorithms. In *Astronomical Data Analysis Software and Systems V*, vol. 101 of *ASP Conference Series*, 41–44.