CLIMP - Cluster-based Imputation of Missing Values in Microarray Data

Nils Gehlenborg
gehlenbo@cs.washington.edu

December 19, 2003

Introduction

Since their invention in the mid-1990s many of improvements have been achieved concerning the quality of microarrays. Different kinds of microarrays are in use today in many fields, which has led to a vast number of preprocessing and analysis techniques for data from such microarrays. Due to their complexity and high sensitivity to all different kinds of influences during manufacturing and experiment, it is often impossible to determine the expression level of all genes on the microarray. Genes for which no expression level could be determined will be missing values in in the subsequent data preprocessing and analysis steps.

The reasons why values are missing in microarray data are diverse. It is possible to distinguish two classes of error that lead to missing values. First, values might be missing arbitrarily, for example because the spot intensity could not be measured or because the background corrected spot intensity turned out to be negative, due to a background signal higher than the actual spot signal. Another problem encountered particularly in cDNA microarrays are spots that have a very low signal in both channels and thus the ratio is not very robust, which leads to the values being declared missing to prevent harmful consequences in the data analysis. While these problems may be due to the actual expression levels in the cell, values might as well be missing due to the handling of the microarray during the experiment. Such problems are, for instance, dust or scratches on the surface of the array. Figure 1 shows such a microarray.

The second class of errors that causes values to be missing are systematical errors. Examples for systematical errors are problems during array production, for instance, when the arrayer[1] used a broken printing tip to spot cDNA on the glass slide.

The consequences of any type of missing value are grave, as most of the standard analysis techniques are not prepared to handle missing values appropriately. These range from such basic computations as similarity or dissimilarity measures between two genes to methods like principal component analysis, singular value decomposition, clustering and classifiers such as support vector machines. As several of these techniques are commonly applied during data analysis, the data must not contain any missing values.

There are different ways to approach the problem of missing values in microarray data. An expensive solution is to use two or more microarrays instead of only one. The more arrays are used, the higher the probability to find at least one value for each gene in each condition. This approach is expensive, not perfect and introduces more complexity into the analysis, in particular if results differ between replicates.

There are two inexpensive but destructive methods that can be applied to handle missing values. *Casewise deletion* excludes all genes that have a missing value in one or more conditions from further steps of the analysis. This is clearly inefficient and in the worst case all genes are excluded. In the *pairwise deletion* method genes are not completely removed, but whenever a distance or another pairwise value is computed, all missing conditions are left out in both genes. As this is the same as defining the values in those conditions being the equal, distances tend to become smaller for genes with missing values.

Besides these destructive approaches, there are a series of constructive techniques to estimate the missing values. The most simple one is it to fill in zeros (in the case of log-ratios) [1] or ones (in the case of ordinary ratios) for missing values. Alternatively, row or column averages can be used to replace missing values.

The weakness of these techniques is that they do not take into account the correlation structure of the expression data [2]. A few methods have been proposed that consider this information to estimate missing values. Troyanskaya *et al.* [2] describe a technique based

[1]The robotic device used to manufacture cDNA microarrays.

on singular value decomposition (SVDimpute) and another one based on the *k* nearest neighbors algorithm (KNNimpute). Comparing their methods with filling in of zeros and row averages they found that the KNNimpute algorithm outperforms all other methods, particularly when genes are expressed in small clusters.
Two approaches based on Bayesian inference were published recently. The one developed by Oba *et al.* [3] employs Bayesian principal component analysis (BPCA), which uses principal component regression, Bayesian estimation and the expectation-maximization (EM) algorithm to estimate the missing values. Another method was described by Zhou *et al.* [4] that uses linear or non-linear regression with Bayesian gene selection to find genes that are highly correlated to a target gene, which in consequence are used for missing value estimation in the target gene. Both papers using Bayesian methods found that their respective method works better than KNNimpute.
In this paper a method is proposed that uses hierarchical clustering to identify correlated genes.

System and Methods

Starting points

There are several possible starting points for missing value estimation in microarray data. The most information rich is the actual image of the scanned microarray. An extract of such an image is shown in Figure 1. Given the image of the microarray and information about the spots it is possible to identify systematical errors and in some cases also the reason why a value is missing. For instance, if there is no signal in many or all spots in a particular region of the microarray, it was obviously damaged or subject to incorrect handling. In such cases it seems wise to repeat the experiment.
The main problem with the image as a starting point is the high complexity involved in processing the image file, and different types of images obtained from different types of microarrays. A second starting point is the output of the image analysis program. With this data it is no longer possible to identify the reason why a value is missing, but it is still possible to identify systematic errors. However, different image analysis programs will output different parameters and the output for a oligo microarray will be significantly different from the output for a cDNA array. Identification of systematical patterns of missing values is useful to decide whether it makes sense to estimate missing values or if the experiment should be repeated since

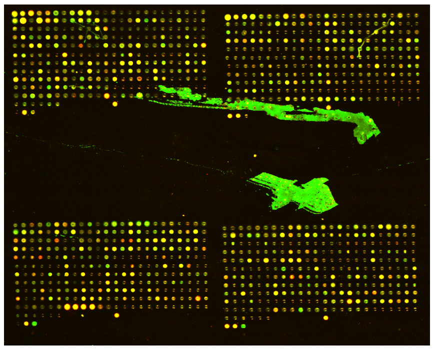

Figure 1: Extract of an image from a scanned cDNA microarray. The two bright green areas in the center of the image cover up a part of the spots on the array, which are likely to be declared missing. There are also traces of dust visible on the image. The original image was taken from the Stanford Microarray Database (http://genome-www.stanford.edu/microarray).

there is strong evidence for a defect microarray.
The expression matrix contains the least information of all starting points mentioned, and thus the detection of systematical errors is very limited if an expression matrix is used as starting point. But it is the most common format[2] and applicable for all types of microarrays. Therefore it is the most favorable choice when missing values are to be estimated in data from different types of microarrays.

CLIMP algorithm

The CLIMP (cluster-based imputation) algorithm described below performs missing value estimation on an expression matrix[3]. Given an expression matrix where rows represent genes and columns represent conditions, the algorithm will estimate all missing values in the matrix. Three requirements have to be fulfilled by the expression matrix. First, columns without any non-missing values are not allowed[4], second, rows with 50 % or more missing values are not allowed and finally, there must be at least one row without any missing value. CLIMP employs the correlation structure of gene expression data to estimate missing values.
The basic steps of the algorithm are described in the following. Details are explained later. Figure 2 shows a symbolic expression matrix. Each bar represents a gene and each square in a bar represents a condition. The white square represents a missing value that is to

[2]This might well change when standards like MIAME [6] become better supported by analysis software.
[3]All other methods listed in the introduction use expression matrices as well.
[4]The implementation requires less than 90 % missing data per column.

be estimated by the algorithm. For simplicity there is only one missing value in this matrix, but the algorithm is able to handle an almost arbitrary number of missing values.

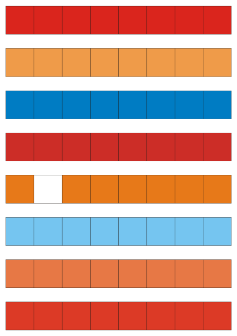

Figure 2: An expression matrix with a single missing value. Each bar represents a gene and each column represents a condition. The color of the bars represents a particular expression profile. There are three different types of expression profiles in the expression matrix. The different shades of red, orange and blue represent slight variations in the expression profiles. The red and the orange expression profiles are pretty similar (hence the similar colors), but the blue one is the inverse of the orange one (hence the complementary colors). White represents missing values.

To find genes that are similar to the gene with the missing value, CLIMP clusters the gene expression matrix using a complete-linkage hierarchical clustering algorithm. Complete-linkage is chosen because the method tends to produce dense clusters. Figure 3 displays the hierarchy found by the clustering algorithm and the (sorted) expression matrix.

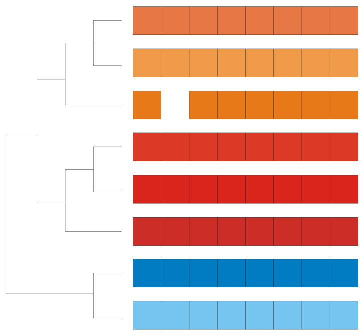

Figure 3: Hierarchical clustering applied to the expression matrix. It it is not clear whether there are three clusters (orange, red, blue) or only two (orange and red, blue).

To determine the number and composition of clusters, CLIMP extracts all clusters with a size below a threshold value from the hierarchy. This is illustrated in Figure 4. Finally, the missing value is computed as a ranked distance weighted average of all genes in the cluster of the gene with the missing value.

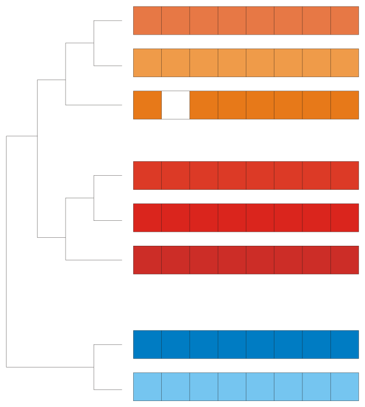

Figure 4: Clusters for an upper threshold of 5. Note that for a threshold of 6 there would be only two clusters.

Details of the CLIMP algorithm

Often there are many distinct patterns of missing values in an expression matrix. The number of distinct patterns is correlated with the number of missing values. Each distinct pattern is a so-called *pattern of missingness* (POM). Each POM is a set of columns. The length of a POM is the cardinality of the set of columns defining the POM. An expression matrix with several different POM is shown in Figure 5.

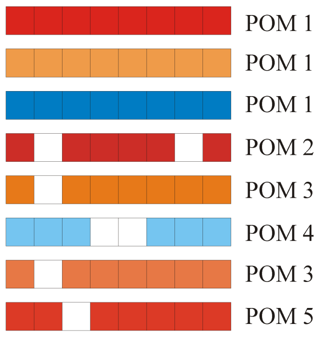

Figure 5: An expression matrix with five distinct POM. The lengths of the POM are 0 (POM 1), 1 (POM 3, POM 5) and 2 (POM 2, POM 4).

Before CLIMP clusters the expression matrix, it identifies all distinct POM and the rows of the expression matrix associated with them. The rows associated with the POM of length 0 (which exist by definition) are referred to as the *base matrix*. The base matrix is the maximal complete submatrix of the expression matrix and used for estimation of missing values (in all rows not part of the base matrix).
The clustering is performed individually for each POM. A *work matrix* is created from the rows of the base matrix and the rows associated with the POM.

All columns defining the POM are removed from the *work matrix* and the clustering algorithm is applied to this matrix. For the expression matrix shown in Figure 5 the clustering algorithm has to be run on the four work matrices shown in Figure 6.

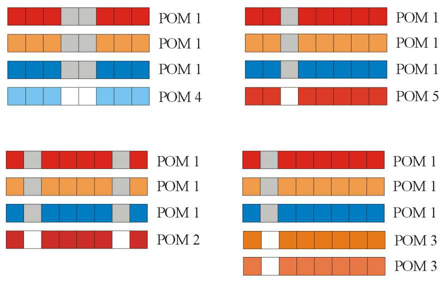

Figure 6: Four different work matrices. Gray, respectively white squares are removed before clustering. Rows with gray squares represent the base matrix.

As mentioned above, the clusters are determined by an upper threshold for the cluster size. But as some genes might be outliers and thus form a cluster of size 1 and to rule out too small clusters in general, a lower threshold is defined. If a cluster size is below this lower threshold, estimation of missing values is based on a k nearest neighbors approach.

Results and Discussion

The CLIMP algorithm was implemented in R and compared to the KNNimpute[5] and the BPCA[6] algorithm.

The test data set was based on data published by Spellman *et al.* [5]. Two of the four time-series experiments performed to analyze the yeast cell cycle were used for the comparison. The data sets differ in the way how the yeast cells were synchronized and are labeled α-factor and *cdc15* for the corresponding synchronization methods. The α-factor expression matrix comprises 18 conditions (time points) and the *cdc15* expression matrix comprises 15 conditions (time points). Both matrices contained about 6100 yeast open reading frames.

From each of the two matrices the maximal complete submatrix was extracted. Based on the submatrices six matrices with different patterns of missing values were created at levels of 1 %, 2 %, 5 % and 10 % missing values. The values were removed using the rate-based method described in [3]. In consequence, a total of $6 \times 4 = 24$ matrices with missing values was created from each of the two complete submatrices obtained from the α-factor and the *cdc15* matrices.

The performance of the three methods was evaluated using the normalized root mean squared error (*NRMSE*) as proposed by Troyanskaya *et al.* and as used in the papers by Oba *et al.* and Zhou *et al.*. The *NRMSE* is defined as

$$NRMSE = \sqrt{\frac{\text{mean}((O-E)^2)}{\text{variance}(O)}},$$

with O being the original matrix and E being the estimated matrix[7]. The closer *NRMSE* to 0 (zero), the more accurate the estimation. If $NRMSE = 0$ it follows that $O = E$. The closer *NRMSE* to 1, the worse the estimation.

Each of the three algorithms was run on subsets of size 100, 500, 1000 and 2000 on each of the 24 matrices for each data set. Thus it is possible to analyze the effect of the size of the data set on the estimation accuracy. For each percentage of missing values and for each subset size the *NRMSE* was averaged over the six matrices[8].

The parameter chosen for CLIMP were 35 as maximal cluster size and 20 as minimal cluster size. The k chosen for KNNimpute and the k nearest neighbors estimation in CLIMP was $k = 17$, as Troyanskaya *et al.* had found that this is in the range of values for k that yields best results. The distance measure used in both KNN and CLIMP was the Euclidean distance as proposed by Troyanskaya *et al.*. The BPCA algorithm does not require the input of any parameters, as the selection of all model parameters is part of the algorithm.

The results for the α-factor data set are shown in Figure 7, the results for data set *cdc15* are presented in Figure 8. The figures show that CLIMP has almost identical performance as KNNimpute. BPCA performs much better than the other two methods, which confirms the findings of Oba *et al.*. The plots indicate that missing values in larger expression matrices are usually estimated with higher accuracy, particularly for the BPCA algorithm. However, the more data is missing, the less accurate the estimation.

It remains open how often CLIMP used its integrated k nearest neighbor algorithm for estimations in these

[5]The R implementation in the `pamr` package was used for this comparison.

[6]The Java implementation available on the website of the authors was used for this comparison.

[7]Here the complete matrices were used, it is also possible to compute the *NRMSE* only on the missing, respectively estimated values.

[8]KNNimpute was unable to estimate missing values in three cases where the subset size was 100.

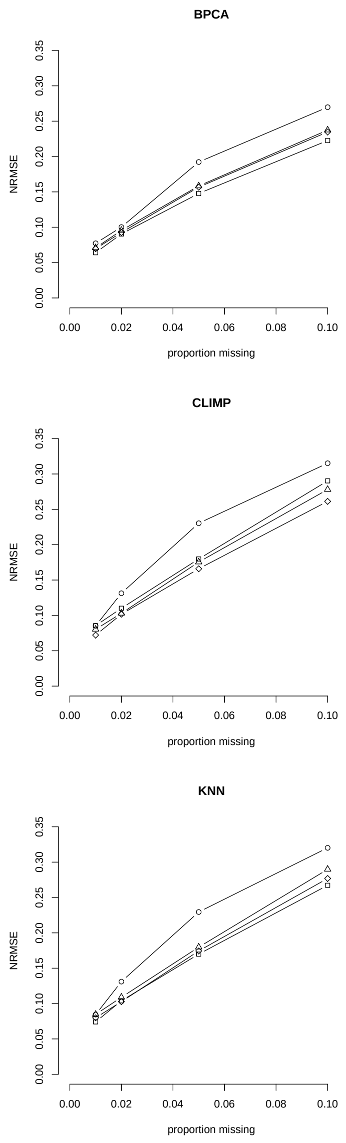

Figure 7: Results of BPCA, CLIMP and KNNimpute on the matrices created from the α-factor data set. The results of CLIMP and KNNimpute are almost identical. Each of the four lines in the plots represents a subset size, circles represent 100, triangles 500, squares 1000 and diamonds 2000 genes.

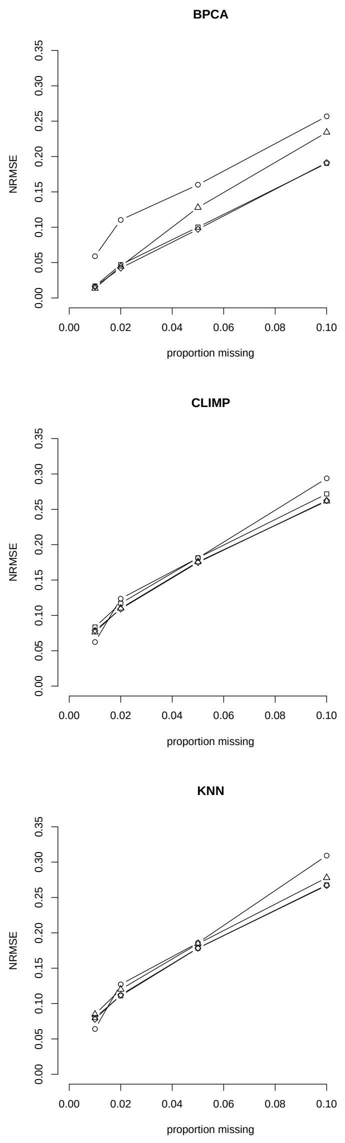

Figure 8: Results of BPCA, CLIMP and KNNimpute on the matrices created from the *cdc15* data set. The results of CLIMP and KNNimpute are almost identical. Each of the four lines in the plots represents a subset size, circles represent 100, triangles 500, squares 1000 and diamonds 2000 genes.

experiments. The use of *k* nearest neighbor is presumably more frequent when the expression matrix has only few genes, because the clustering algorithm will not be able to identify dense clusters. Knowledge about how often *k* nearest neighbors is used in CLIMP would be helpful in determining how well CLIMP actually performs. Intuitively CLIMP should perform better than KNN, as it allows the number of genes used for estimations to be selected in a way more appropriate to how the data is generated.

However, the multiple clustering steps in CLIMP make the algorithm quite slow. On the other hand, this is not a problem, since time is not an important issue in missing value estimation. This step is performed only once during the preprocessing of the microarray data and even if it takes a day or two, this will be little time compared to the time spent on experimental work in the lab and the subsequent data analysis.

A more serious shortcoming of the current implementation of CLIMP is that the algorithm always requires at least one row without missing values in the expression matrix. This problem may be overcome by implementation of a method that creates a base matrix from two disjunct POM that have a large number of rows associated with them and only a small number of missing conditions. Conditions missing in one block of rows would be estimated using the respective other one.

It is also questionable if the values chosen as upper and lower bound for the cluster size are ideal. If these parameters are tuned, the performance of CLIMP might increase. The same holds for the ranked distance weighted average currently used to estimate missing values, it would probably be more appropriate to use weights based on the actual similarity or distance and not on their rank.

CLIMP might gain more prediction accuracy if the base matrix would be extended after each POM is processed. The rows associated with the processed POM could be added to the base matrix, which would yield clusters with more similar expression profiles. It is not clear whether estimated values should be used to estimate other missing values, but even if this is not done, more than 50 %[9] of the values in each row can be employed in estimations of missing values. Ideally, POM are processed in order of increasing length. Alternatively, it is possible to reestimate all missing values in a two-stage process. In the first stage the algorithm is applied as describe before and in the second stage the complete estimated matrix is used as base matrix for reestimation of the missing values. This would also have a beneficial effect on the clustering.

The accuracy of the estimations might also be increased if the properties of the values used to estimate a missing value are considered. Figure 9 illustrates three different scenarios that might occur.

In the topmost case, the estimation will probably be very accurate, as the values for the estimation are all very similar. In the second case though, the values form two clusters, one of them above, the other below zero. The average would be a value close to zero. But it seems not very likely that the true value was close to zero, if we consider genes with similar expression profiles. In that case it would make sense to identifiy subclusters within the genes selected for the estimation and use only genes from the subcluster closer to the target gene. The third case is the worst, identification of subclusters will not yield any benefit. Generally, the described subclustering step will require a large base matrix to find meaningful subclusters.

[9]The CLIMP algorithm requires that at least 50 % of the values in each row are non-missing.

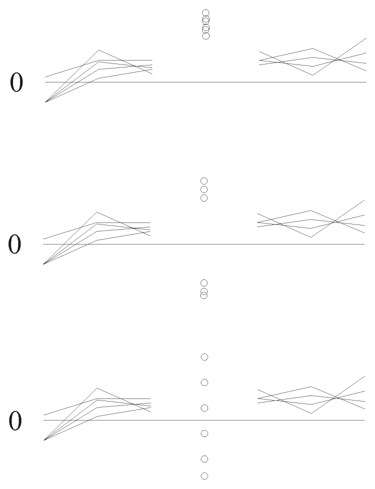

Figure 9: Three scenarios for values used to estimate a missing values. The data shown represent one cluster obtained by hierarchical clustering.

Conclusions

CLIMP is an algorithm with performance equal to that of KNNimpute and there are several unimplemented features that might improve the estimation accuracy of CLIMP. However, it seems very likely that the algorithms based on Bayesian inference will still be the best choice, as their accuracy is significantly better than that of KNNimpute and CLIMP.

References

[1] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, and X. Yu, *Nature* **403**, 503-511 (2001).

[2] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman, *Bioinformatics* **17**, 520-525 (2001).

[3] S. Oba, M. Sato, I. Takemasa, M. Mondern, K. Matsubara, and S. Ishii, *Bioinformatics* **19**, 2088-2096 (2003).

[4] X. Zhou, X. Wang, and E. R. Dougherty, *Bioinformatics* **19**, 2302-2307 (2003).

[5] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, *Molecular Biology of the Cell* **9**, 3273-3297 (1998).

[6] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. P. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron, *Nature* **29**, 365-371 (2001).