Missing Value Estimation In DNA Microarray – A Fuzzy Approach

Sumit Chakraborty
Lecturer, CS Dept
Ramsaday College, Howrah,WB
sm.chakraborty09@gmail.com

Sujay Saha
Asst. Professor, CSE Dept
Heritage Institute of Technology
sujay.saha@heritageit.edu

Kashinath Dey
Associate Professor, CSE Dept
University of Calcutta
kndey55@gmail.com

Abstract **- DNA microarray technology which is used in molecular biology, allows for the observation of expression levels of thousands of genes under a variety of conditions. The analysis of microarray data has been successfully applied in a number of studies over a broad range of biological disciplines. Now it is very unfortunate that various microarray experiments generate data sets containing missing values. Since most of the algorithms for gene expression analysis require a complete gene array as input, the missing values need to estimate. The methods exist for estimating missing values are like KNNimpute, SVDimpute, LLSimpute, LLS-SVDimpute etc. In this paper we present a new fuzzy technique Fuzzy Difference Vector Impute (FDVimpute) for estimating missing values in a DNA microarray.**

Keywords – **DNA microarray, missing value, LLS-SVDImpute, FDVImpute, Spellman dataset**

I. INTRODUCTION

Gene expression microarrays provide a popular technique to monitor the relative expression of thousands of genes under a variety of experimental conditions. In spite of the enormous potential of this technique, there remain challenging problems associated with the acquisition and analysis of microarray data that can have a profound influence on the interpretation of the results.

Gene expression microarray experiments can generate data sets with multiple missing expression values [1]. Unfortunately, many algorithms for gene expression analysis require a complete matrix of gene array values as input. Methods such as hierarchical clustering and K-means clustering are not robust to missing data, and may lose effectiveness even with a few missing values. Methods for imputing missing data are needed, therefore, to minimize the effect of incomplete data sets on analyses, and to increase the range of data sets to which these algorithms can be applied. There are several ways to deal with missing values such as deleting genes with missing values from further analysis, filling the missing entries with zeros, or imputing missing values of the average expression level for the gene

Missing values can lead to erroneous conclusions about data and substitution of missing values may introduce inaccuracies and inconsistencies. These values can negatively impact discovery results, and errors or data skews can proliferate across subsequent runs and cause a larger, cumulative error effect. As well, most analysis methods cannot be performed if there are missing values in the data. Missing values may also prevent proper classification and clustering.

So the proper and more accurate prediction of Missing values remains an important step on the way to get better results. The goal of missing value is to represent an accurate data set of genes, species, etc. A variety of approaches have been proposed for estimating missing values in DNA microarrays. Some of these methods are very complex and take a lot of time, while other having less accuracy.

The organization of the article is as follows: In section II, we describe some already existing methods for estimating missing values. Section III presents the proposed missing value estimation method. In section IV, we provide experimental results along with comparisons. Section V discusses about Conclusion & Future scope.

II EXISTING METHODS

There are several methods to estimate missing values in a DNA microarray dataset. Four among them are briefly described as follows:

LLSimpute Algorithm

Local least square imputation method (LLSimpute) represents a target gene that has missing values as a linear combination of similar genes [2]. The similar genes are chosen by k-nearest neighbors or k coherent genes that have large absolute values of Pearson correlation coefficients.

G denote a gene expression data matrix with *m* genes and *n* experiments and assume m > > n (G is a $m \times n$ matrix). In the matrix G, a row g_i represents the i th row with n experiments ($1 \times n$ matrix).

$$G = \begin{pmatrix} g_1 \\ g_2 \\ . \\ . \\ g_m \end{pmatrix}_{m \times n}$$

A missing value in the l th location of the i th gene is denoted as α, i.e. $G(i, l) = \mathbf{g}_i(l) = \alpha$. There are two steps in the LLSimpute method.

First step is to select k ($k < m$) other genes with expressions most similar to that of **g**1 and with the values in their first positions not missing. The second step is regression and estimation, regardless of how the *k* genes are selected To recover a missing value α in the first location $\mathbf{g}_1(1)$ of $\mathbf{g}_1$ in *G* the *k*-nearest neighbor gene vectors for $\mathbf{g}_1$ are found for LLSimpute. In this process of finding the similar genes, the first component of each gene is ignored following the fact that $\mathbf{g}_1(1)$ is missing. Now there are several metric to choose K similar Genes such as, Euclidian Metric, Pearson Coefficients etc. The LLSimpute using the Pearson Coefficients to select k genes is referred as LLSimpute/PC.

Based on these *k*-neighboring gene vectors, the matrix $A \in R^{k \times (n-1)}$ and the two vectors $b \in R^{k \times 1}$ and $w \in R^{(n-1) \times 1}$ are formed. The *k* rows of the matrix *A* consist of the *k*-nearest neighbor genes $g_{si}, 1 \le i \le k$ with their first values deleted, the elements of the vector **b** consists of the first components of the *k* vectors g_{si} and the elements of the vector **w** are the $n - 1$ elements of the gene vector $\mathbf{g}_1$ whose missing first item is deleted. After the matrix *A* and the vectors **b** and **w** are formed, the least squares problem is formulated as

$$\min_{x} \left\| A^T x - w \right\|$$

Then, the missing value α is estimated as a linear combination of first values of genes

$\alpha = b^T x = b^T (\mathrm{A}^T)^{\dagger} w$ Where $(\mathrm{A}^T)^{\dagger}$ is the pseudo inverse of A^T.

KNNimpute Algorithm

This method is very much same as LLSimpute method except the estimation phase. This method selects genes with expression profiles similar to the gene of interest to impute missing values [1]. If we consider gene A that has one missing value in experiment 1, this method would find K other genes, which have a value present in experiment 1, with expression most similar to A in experiments 2–*N* (where *N* is the total number of experiments). A weighted average of values in experiment 1 from the K closest genes is then used as an estimate for the missing value in gene A. In the weighted average, the contribution of each gene is weighted by similarity of its expression to that of gene A. To recover a missing value α in the first location $\mathbf{g}_1(1)$ of $\mathbf{g}_1$ in *G* the *k*-nearest neighbor gene vectors for $\mathbf{g}_1$ are found for KNNimpute. In this process of finding the similar genes, the first component of each gene is ignored following the fact that $\mathbf{g}_1(1)$ is missing. There are several metric to choose K similar Genes such as, Euclidian Metric.

SVDimpute Algorithm

The SVDimpute method uses Singular Value Decomposition of matrices to estimate the missing values of a DNA micro array [1]. This method works in two steps. First it decomposes the Gene data matrix into a set of mutually orthogonal expression patterns that can be linearly combined to approximate the expression of all genes in the data set. SVD is based on a theorem from linear algebra which says that a rectangular matrix A can be broken down into the product of three matrices - an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V. The theorem is usually presented as follows

$$A_{m,n} = U_{m,m} S_{m,n} V_{n,n}^{T}$$

Let A be a matrix with m rows and n columns. Now the SVD method is applied on this matrix as follows:

At first, AA^T is computed, which is a square (Symmetric) matrix with m rows and m columns. So this matrix has m eigen values (Real no) and m corresponding eigen vectors. Let X be the set of eigen vectors of AA^T, arranged according to their corresponding eigen values (eigen values are sorted in descending order on their absolute values). So the columns of X are eigen vectors of AA^T, Now normalized each columns of X to obtained the orthonormal matrix U. Secondly, let Y be the set of eigen vectors of A^TA, arranged according to their corresponding eigen values (eigen values are sorted in descending order on their absolute values).So the columns of Y are eigen vectors of A^TA, Now normalized each columns of Y to obtained the orthonormal matrix V. Thirdly, compute the absolute values of the eigen values of AA^T or A^TA and construct a diagonal matrix S whose diagonal elements are the square roots of these values sorted in descending order. Once k most significant eigengenes from V^T are selected, the missing value j in gene i can be estimated by first regressing this gene against the k eigengenes & then by using the coefficients of the regression to reconstruct j from a linear combination of the k eigengenes.

LLS-SVDimpute Algorithm

This algorithm is a modification of existing SVDimpute method. Since SVD can only be performed on complete matrices, the original SVDimpute algorithm first applies row average to substitute all missing values in A to obtain A'. After that SVDimpute algorithm is applied on A' for estimating each missing values. But LLS-SVDimpute algorithm [3] is different from the earlier version of SVDimpute in the initial filling of missing cells of a DNA microarray. LLS-SVDimpute algorithm first applies LLSimpute method to estimate the missing cells initially, & then uses SVDimpute algorithm to approximate those values further.

III PROPOSED METHOD

In all the existing methods for estimating missing values in DNA microarray, the most similar genes of the target gene are selected on the basis of certain metrics, like Pearson correlation coefficients, Euclidean distance etc. The genes will be selected if satisfy certain conditions, otherwise ignored. But this is a kind of crisp technique & it doesn't always produce good results. That's why we try to incorporate some fuzziness in the proposed method FDVimpute of estimating missing values.
FDVimpute estimates the missing cells of datasets in two steps. The first step selects nearest (most similar) genes of the target gene (whose some component is missing) using Fuzzy Difference Vector algorithm. Then the missing cell is estimated by using least square feet on the selected genes in the second step.

Selecting Neighboring Genes

Given a dataset G of **m** genes y_1, y_2, …. y_m each of which have **n** dimensions. Now the difference vector V_i of the ith gene y_i is calculated as follows:
$V_i(k) = y_i(k) - y_i(k+1)$, $1 \leq k \leq n-1$
Let's suppose m_{ij} contains the number of matches between the difference vectors V_i and V_j of the respective genes. A match in the kth component of the vectors V_i and V_j is determined by whether the sign of $V_i(k)$ and $V_j(k)$ is same or not. In this way we calculate the total number of matches between the ith and jth genes, that m_{ij} contains. Since it is a fuzzy approach, we define a membership function $\mu(i)$ for i th gene as follows:

$$\mu(i) = m(i)/(n-1)$$

If the membership value of gene i is greater than the predefined threshold Θ then that gene will be selected as the neighboring gene of the target gene.
The following Algorithm selects the most similar genes of the data matrix X with m rows & n columns where the 1st gene is treated as the target gene. That's why in the computation of the difference vectors the 1st component of each gene is ignored.

```
Step 1: For i=1 to m do  // Constructs the difference
                         //   table
            For j=1 to n-1 do
              diff_table (i, j) =X (i, j)-X (i, j+1)
            End
        End
Step 2: For i=2 to m do // Computes the matches for
                        //  membership function
          m (i)=0
          For j=1 to n-1
            if (diff_table(1,j)*diff_table(i,j)>0) then
                m (i)++
            End If
          End
        End
        k=0
Step 3: For i=2 to m do   // Selects the nearest genes
            μ (i) = m(i)/(n-1)
            If μ(i)>Θ then
                k=k+1
              For j=1 to n do
                A(k,j)=X(i,j)   // A contains most similar
                                // genes of target
            End
          End If
        End
```

The time complexity of the above algorithm is O (m*n).

Formulation of Least Square feet

Based on these *k* gene vectors of $A_{k,n}$, the matrix $B \in R^{k\times(n-1)}$ and the two vectors $b \in R^{k\times 1}$ and $w \in R^{(n-1)\times 1}$ are formed. The *k* rows of the matrix B consist of the *k* genes of A with their first values deleted, the elements of the vector **b** consists of the first components of the *k* vectors of A and the elements of the vector **w** are the *n* − 1 elements of the gene vector $\mathbf{g}_1$ whose missing first item is deleted. After the matrix *A* and the vectors **b** and **w** are formed, the least squares problem is formulated as

$$\min_{x} \left\| B^T x - w \right\|$$

Then, the missing value *α* is estimated as a linear combination of first values of genes

$$\alpha = b^T x = b^T (\mathrm{B}^\mathrm{T})^{\dagger}\ w$$

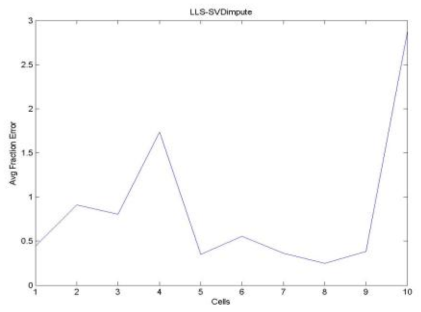
Where $(\mathrm{B}^\mathrm{T})^{\dagger}$ is the pseudo inverse of B^T.

$$\therefore \alpha = \mathrm{b}^\mathrm{T} x = x_1 b_1 + x_2 b_2 + \cdots + x_\kappa b_\kappa ,$$

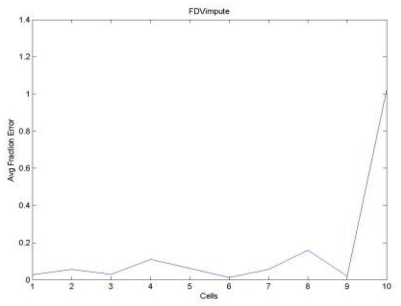
IV EXPERIMENTAL RESULTS

We have applied our FDVimpute algorithm on ‘Spellman’ data set, having 6025 genes and approximately 76 experiments in each gene. The algorithm is coded with MATLAB 6.5 & run on an Intel Pentium IV computer with 3.06 GHz Processor and 256 RAM. The parameters ‘Error’ & ‘Fraction_eror’ will be calculated by |Estimated_value – Original_value| and |Error/Original_value| respectively. The performance measure of various methods depend on the parameter ‘Fraction_eror’, as it lowers the accuracy is better.

Since we know that LLS-SVDimpute method shows better results than earlier KNN & SVDimpute methods in most of the cases, the following table TABLE 1 displays the comparative results for some of the experiments between the proposed FDVimpute & LLS-SVDimpute.



The Comparison of results are shown using performance diagrams drawn on the basis of the parameter ‘Average Fraction Error’ as follows:



V CONCLUSION

Considering the above results and diagrams of the experiments done on the “Spellman” dataset we can conclude that FDVImpute produces better results than LLS-SVDImpute algorithm. So, this proposed method is so far best among all the methods exist to estimate missing values of a DNA microarray. Since FDVImpute is a Fuzzy approach, so the performance can be improved by taking a more efficient membership function µ in the selection algorithm and the appropriate choice of the threshold Θ.

TABLE 1

Comparison of the Results of the proposed FDVimpute method with LLS-SVDimpute

			LLS-SVDimpute Result						FDVimpute Result				
Row	Col	Ori_val	K(LLS)	K(SVD)	Esti_val	Error	Frac_eror	Avg. Fraction Error	Θ	Esti_val	Error	Frac_eror	Avg. Fraction Error
1	10	-0.4400	200	30	-0.2803	0.1597	0.3629	0.4645	0.3000	-0.4214	0.0186	0.0422	0.0273
			200	40	-0.2747	0.1653	0.3758		0.4000	-0.4276	0.0124	0.0282	
			300	50	-0.2072	0.2328	0.5291		0.5000	-0.4337	0.0063	0.0143	
			100	40	-0.2546	0.1854	0.4213		0.6000	-0.4280	0.0120	0.0274	
			100	50	-0.1614	0.2786	0.6333		0.7000	-0.4293	0.0107	0.0244	
2	5	-0.1100	300	30	0.1595	0.2695	2.4502	1.7685	0.3000	-0.1122	0.0022	0.0197	0.1101
			300	40	0.0246	0.1346	1.2233		0.4000	-0.1147	0.0047	0.0429	
			200	30	0.1571	0.2671	2.4280		0.5000	-0.1190	0.0090	0.0816	
			200	40	0.0210	0.1310	1.1910		0.6000	-0.1332	0.0232	0.2112	
			200	50	0.0605	0.1705	1.5502		0.7000	-0.1314	0.0214	0.1950	
5	51	0.5300	200	30	0.6491	0.1191	0.2247	0.3478	0.3000	0.5111	0.0189	0.0356	0.0618
			200	50	0.2742	0.2558	0.4826		0.4000	0.5097	0.0203	0.0383	
			300	30	0.6548	0.1248	0.2354		0.5000	0.5036	0.0264	0.0498	
			300	40	0.3457	0.1843	0.3477		0.6000	0.4952	0.0348	0.0657	
			100	40	0.3519	0.1781	0.3360		0.7000	0.4881	0.0419	0.0791	
			100	50	0.2860	0.2440	0.4603		0.8000	0.4757	0.0543	0.1024	
27	8	0.2800	200	30	0.1275	0.1525	0.5448	0.3618	0.3000	0.2700	0.0100	0.0357	0.0570
			200	40	0.2498	0.0302	0.1080		0.4000	0.2708	0.0092	0.0329	
			300	40	0.2502	0.0298	0.1065		0.5000	0.2700	0.0100	0.0356	
			300	50	0.1461	0.1339	0.4781		0.6000	0.2585	0.0215	0.0767	
			100	30	0.1359	0.1441	0.5145		0.7000	0.2567	0.0233	0.0832	
			100	50	0.1626	0.1174	0.4193		0.8000	0.2582	0.0218	0.0780	
501	76	0.5400	200	30	0.3473	0.1927	0.3569	0.3834	0.3000	0.5272	0.0128	0.0237	0.0208
			200	40	0.2491	0.2909	0.5388		0.4000	0.5273	0.0127	0.0235	
			300	40	0.2479	0.2921	0.5409		0.5000	0.5261	0.0139	0.0257	
			300	50	0.6755	0.1355	0.2510		0.6000	0.5308	0.0092	0.0170	
			100	30	0.3490	0.1910	0.3537		0.7000	0.5307	0.0093	0.0173	
			100	50	0.6798	0.1398	0.2588		0.8000	0.5306	0.0094	0.0174	

VI REFERENCES

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

[2] H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: local least squares imputation", Bioinformatics, 21, 187 – 198, 2005

[3] S. Chakraborty, S. Saha, K.N.Dey, "Modified SVDImpute Algorithm for Estimating Missing Values in a DNA Microarray", IEEE EDS Student Conference 2011, Jointly Organized by HIT-K & IEEE EDS Calcutta Section, April 2011

[4] O. Alter, P. O. Brown, and D. Botstein "Singular value decomposition for genome-wide expression data processing and Modeling". *Proc. Natl Acad. Sci. USA*, **97**, 10101–10106, 2000.

[5] G. H. Golub and C. F. van Loan "*Matrix Computations"*; 3rd edn. Johns Hopkins University Press, Baltimore, CA, 1996.

[6] M. P. Brown, W.N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares,Jr. and D. Haussler (2000) "Knowledge based analysis of microarray gene expression data by using support vector machines". *Proc. Natl Acad. Sci. USA*, **97**, 262–267, 2000.

[7] S. Raychaudhuri, J. M. Stuart and R. B. Altman "Principal components analysis to summarize microarray experiments:application to sporulation time series". *Pa Symp. Biocomput.*, 455–466, 2000.

[8] G. N. Wilkinson "Estimation of missing values for the analysis of incomplete data". *Biometrics*, **14**, 257–286, 1958.

[9] Y. Yates "The analysis of replicated experiments when the field results are incomplete." *Emp. J. Exp. Agric.*, **1**, 129–142, 1933