Kernel PCA Regression for Missing Data Estimation in DNA Microarray Analysis

Ying Shan and Guang Deng
Department of Electronic Engineering
La Trobe University
Bundoora, Victoria, Australia 3086
Email: y.shan,d.deng@latrobe.edu.au

***Abstract*— The DNA microarray may contain missing expression data. Estimation of missing values is a necessary step in microarray analysis, because data mining procedures require a complete expression as their input. In this paper, we propose a missing data estimation algorithm, named *KPCAimpute*, based on kernel principal component analysis. We consider a family of heavy-tailed kernel functions, which is a generalization of the famous Gaussian kernel. The performance of the proposed KPCAimpute algorithm is compared with two state-of-the-art linear regression methods, i.e., Bayesian principal component analysis imputation (BPCA) and local least squares imputation (LLSimpute). The KPCAimpute outperforms the LLSimpute when the missing percentage increases. The performance of the KPCAimpute is similar to that of the BPCA imputation. Therefore, it is an effective and promising algorithm in estimating missing values for DNA microarray profiles.**

I. Introduction

The study of high throughput DNA microarray expression [1] has become an increasingly active research area in bioinformatics, due to the extensive usage of microarrays in biological and clinical applications. Microarray data is generated from a DNA experiment which may contain missing values. Most data mining algorithms such as hierarchical clustering require a complete gene expression as their input. Therefore, the estimation of missing values, also known as data imputation, is a key step in the whole microarray data analysis procedure.

In microarray data imputation literature, linear regression has been used. The K-nearest neighbors imputation (KNNimpute) [2] and sigular value decompostion imputation (SVDimpute) [2] are two typical examples. Recently developed techniques include local least squares imputation (LLSimpute) [3] and Bayesian principal component analysis imputation (BPCA) [4]. LLSimpute uses the K nearest neighbors as the regressors. Regression coefficients are determined in the least squares sense. The BPCA imputation employs eigen samples as the linear regressors, whose presence is governed by corresponding hyperparameters in a Bayesian hierarchical framework.

Kernel principal component analysis (kernel PCA) [5] is a generalization of standard PCA. It effectively exploits the "kernel trick" to find the features of observation data. In kernel PCA regression, the regressors are not the observation data in the input space, but a nonlinear mapping of the observation data into the feature space. The regressors are thus called *the features* of the observation data. However, the nonlinear mapping is often unknown in practice. To avoid this difficulty, a kernel function is defined in the input space. As such, a kernel matrix can be generated, of which each element is defined by the kernel function. The standard PCA is performed on the kernel matrix such that the principal components of the features are first determined. They can be used as the regressors. Kernel PCA has been recently used in DNA microarray analysis, for example, in applications of gene expression data classification [6] and clustering [7] problems.

In this paper we study the kernel PCA method in DNA microarray missing data estimation and propose the KPCAimpute algorithm. In section II, after a brief review of LLSimpute and BPCA, we present the KPCAimpute method. In section III, we present simulation results of the KPCAimpute algorithm and compare it to that of BPCA and LLSimpute. A summary is presented in section IV.

II. Methods

A. Least squares-based imputation and LLSimpute

A linear regression model for microarray imputation can be stated as follows,

$$\mathbf{g} = \mathbf{G}\mathbf{w} + \mathbf{e}, \tag{1}$$

where $\mathbf{g}$ represents an $(N \times 1)$ gene expression, $\mathbf{G}$ is an $(N \times M)$ expression matrix whose columns correspond to various genes and rows to experiments, $\mathbf{w}$ is an $(M \times 1)$ coefficient vector and $\mathbf{e}$ is an $(N \times 1)$ vector representing noise. The underlying assumption is thus that a gene can be expressed as a linear combination of a set of peer genes.

Without loss of generality, we assume that the first n elements of $\mathbf{g}$, denoted by $\mathbf{g}_n$, are missing. The rest of the vector $\mathbf{g}$ is denoted by $\mathbf{g}_p$. Similarly, the expression matrix $\mathbf{G}$ can be divided into two sub-matrices, $\mathbf{G}_m$ corresponding to an $(n \times M)$ matrix, and $\mathbf{G}_p$ an $(N-n) \times M$ matrix. Therefore, the linear regression model with missing entries can be stated as

$$\begin{bmatrix} \mathbf{g}_n \\ \mathbf{g}_p \end{bmatrix} = \begin{bmatrix} \mathbf{G}_n \\ \mathbf{G}_p \end{bmatrix} \mathbf{w} + \mathbf{e}. \tag{2}$$

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

The vector $\mathbf{w}$ is determined by solving the least squares problem,

$$\mathbf{w} = \arg\min_{\mathbf{w}} ||\mathbf{g}_p - \mathbf{G}_p\mathbf{w}||_2 = (\mathbf{G}_p^T\mathbf{G}_p)^{-1}\mathbf{G}_p^T\mathbf{g}_p. \quad (3)$$

The missing values are then estimated by

$$\hat{\mathbf{g}}_n = \mathbf{G}_n\mathbf{w}. \quad (4)$$

The LLSimpute is an implementation of equations (3) and (4), except that the matrix $\mathbf{G}$ is composed of nearest neighborhood genes of $\mathbf{g}$ measured by the Euclidean distance or Pearson's correlation coefficients [3]. One disadvantage of LLSimpute, however, is that the optimal number of neighbors is determined by a "heuristic" search which increases the computational cost of the algorithm.

B. Linear PCA-based imputation and BPCA

The LLSimpute can be generalized by using principal component analysis (PCA). The singular value decomposition of the covariance matrix $\mathbf{C} = \mathbf{G}\mathbf{G}^T$ is

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \quad (5)$$

where columns of $\mathbf{V}$, denoted by $\mathbf{v}_l$, are called eigenvectors, and diagonal elements of Λ, denoted by λ_l are called eigenvalues of $\mathbf{C}$. The principal axes of $\mathbf{G}$ are given by

$$\mathbf{u}_l = \sqrt{\lambda_l}\mathbf{v}_l, \quad (6)$$

where $l = 1 \cdots L$ indicate the indices of the L largest eigenvalues ($L \leq N$). Other axes corresponding to small eigenvalues are considered to be the noise in the input matrix $\mathbf{G}$. Denote $\hat{\mathbf{G}}$ as the principal components of $\mathbf{G}$, which is the projection of $\mathbf{G}$ onto the principal axes. Thus we can approximate the original observation $\mathbf{G}$ by $\hat{\mathbf{G}}$,

$$\hat{\mathbf{G}} = \mathbf{U}^T\mathbf{G}, \quad (7)$$

where columns of $\mathbf{U}$ are the principal axes $\mathbf{u}_l$. After finding the regressors $\hat{\mathbf{G}}$, the regression coefficients can be decided in a similar manner as in equations (3) and (4), where $\mathbf{G}$ is replaced by the counterpart $\hat{\mathbf{G}}$.

A further generalization is to use the Bayesian approach with PCA. The method is called BPCA. The main idea of the BPCA algorithm is to specify the probability density funtions (pdf) about the coefficients vector $\mathbf{w}$. In practice, the pdf are chosen as i.i.d. zero-mean Gaussian distributions, i.e., $w_i \sim N(0, \frac{1}{\alpha_i})$ where $\frac{1}{\alpha_i}$ is the variance and α_i is called a hyper-parameter of the model. The hyper-parameters in the pdf are estimated by using evidence maximization [8]. Under such a probability model, once the hyper-parameters are determined, the pdf of $\mathbf{w}$ are also determined. For example, a very large value of α_i results in a prior for w_i which is sharply peaked at zero. The value of w_i can be regarded as zero. This simply leads to a regressor with w_i being absent from the model. Therefore, using a Bayesian PCA model, the presence or absence of a principal component is governed by the corresponding hyper-parameter estimated from the training data.

1) Kernel PCA in the input space: We review the standard PCA from the kernel function point of view. A kernel function defined in the input space is the inner product of the input $\mathbf{G}$. It is an $(M \times M)$ matrix

$$K = \mathbf{G}^T\mathbf{G}, \quad (8)$$

while the covariance matrix $\mathbf{C} = \mathbf{G}\mathbf{G}^T$ is the result of the outer product of $\mathbf{G}$. It has been pointed out [9] that if the eigenvectors and eigenvalues of K are known, then the principal components of the observation matrix $\mathbf{G}$ can be easily obtained. Indeed, suppose the matrix K has the singular value decomposition

$$K = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^T, \quad (9)$$

we can easily show that the normalized eigenvectors corresponding to the L largest eigenvalues of $\mathbf{C}$ are given by

$$\mathbf{U} = \mathbf{G}\widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}^{-\frac{1}{2}}. \quad (10)$$

Thus, the principal components of $\mathbf{G}$ can be obtained by

$$\hat{\mathbf{G}} = \mathbf{U}^T\mathbf{G} = \widetilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\widetilde{\mathbf{U}}^T\mathbf{G}^T\mathbf{G}, \quad (11)$$

which results in the same form as equation (7).

C. Feature Space Kernel PCA imputation

1) Kernel PCA in the feature space: In this paper, we assume that the missing values may not necessarily depend on observed genes in a linear manner, but rely on a nonlinear regression of peer genes instead. This leads to the following model,

$$\mathbf{g} = \Phi(\mathbf{G})\mathbf{w} + \mathbf{e}, \quad (12)$$

where $\Phi(\mathbf{G})$ denotes a nonlinear mapping from the input space χ ($\mathbf{G} \in \chi$) to the feature space F ($\Phi(\mathbf{G}) \in F$). This model can be regarded as a generalization of equation (1) which is the underlying model for both the BPCA and LLSimpute. A difficulty in equation (12) is that we do not explicitly know the funtion $\Phi(\mathbf{.})$ in practice. This problem is avoided by using the so-called "kernel trick" [5].

We can understand a kernel matrix as the inner product in the feature space

$$\mathbf{K} = \Phi^T(\mathbf{G})\Phi(\mathbf{G}). \quad (13)$$

Note that in practice the kernel matrix $\mathbf{K}$ is conveniently defined in the input space. In fact, the matrix $\mathbf{K}$ can be defined in many different ways subject to the Mercer's condition [5]. Once $\mathbf{K}$ is known, then the underlying nonlinear mapping $\Phi(\mathbf{G})$ is also known.

Performing standard PCA on the kernel matrix,

$$\mathbf{K} = \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T, \quad (14)$$

we can derive similar results as in subsection II-B.1. All we need to do is to replace $\mathbf{G}$ by $\Phi(\mathbf{G})$. Similar to equation (10), the principal axes of $\Phi(\mathbf{G})$ are given by

$$\mathbf{Y} = \mathbf{\Phi}(\mathbf{G})\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}^{-\frac{1}{2}}. \quad (15)$$

Thus, the principal components of $\Phi(\mathbf{G})$ are given by

$$\begin{aligned}\hat{\Phi}(\mathbf{G}) &= \mathbf{Y}^T\mathbf{\Phi}(\mathbf{G}) &(16)\\ &= \hat{\mathbf{\Lambda}}^{-\frac{1}{2}}\hat{\mathbf{U}}^T\Phi^T(\mathbf{G})\Phi(\mathbf{G}) &(17)\\ &= \hat{\mathbf{\Lambda}}^{-\frac{1}{2}}\hat{\mathbf{U}}^T\mathbf{K}. &(18)\end{aligned}$$

Equation (18) clearly illustrates that the principal components in feature space can be explicitly expressed by the kernel matrix $\mathbf{K}$, its eigenvectors $\widetilde{\mathbf{U}}$ and its eigenvalues $\widetilde{\mathbf{\Lambda}}$, although the nonlinear function $\Phi(\mathbf{.})$ is not given.

2) Kernel PCA imputation: In the proposed KPCAimpute algorithm, we replace the regressors from the features $\Phi(\mathbf{G})$ by its counterpart $\hat{\Phi}(\mathbf{G})$,

$$\mathbf{g} = \hat{\Phi}(\mathbf{G})\mathbf{w} + \mathbf{e}. \quad (19)$$

The regression coefficients can be decided in a similar manner as in equations (2) ˜ (4):

$$\begin{bmatrix} \mathbf{g}_n \\ \mathbf{g}_p \end{bmatrix} = \begin{bmatrix} \hat{\Phi}(\mathbf{G}_n) \\ \hat{\Phi}(\mathbf{G}_p) \end{bmatrix} \mathbf{w} + \mathbf{e}. \quad (20)$$

The least squares estimate of $\mathbf{w}$ is now given by

$$\mathbf{w} = [\hat{\Phi}^T(\mathbf{G}_p)\hat{\Phi}(\mathbf{G}_p)]^{-1}\hat{\Phi}^T(\mathbf{G}_p)\mathbf{g}_p. \quad (21)$$

So, the missing values can be estimated by

$$\hat{\mathbf{g}}_n = \hat{\Phi}(\mathbf{G}_n)\mathbf{w}. \quad (22)$$

III. Results

A. Microarray Datasets and Measurement

Two microarray datasets are used in evaluating the performance of KPCAimpute algorithm. The first dataset (Dataset A) is provided along with BPCA software [4] with the size of 758 genes and 50 samples. The second dataset (Dataset B) is a breast cancer microarray from Stanford Microarray Database (SMD) (http://genome-www5.stanford.edu), with 1213 genes and 49 samples.

To evaluate the effectiveness of different methods, missing values are artificially introduced into an original microarray matrix $\mathbf{G}$. The result of an imputation algorithm is another expression matrix of the same size, denoted by $\mathbf{G}^*$. The quality of the imputation method is then measured by using the normalized root mean squared error (RMSE) defined as follows

$$\text{RMSE} = \left(\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}(\mathbf{G}(n,m) - \mathbf{G}^*(n,m))^2}{\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbf{G}^2(n,m)} \right)^{\frac{1}{2}}, \quad (23)$$

where $\mathbf{G}(n,m)$ and $\mathbf{G}^*(n,m)$ are the nth element in the mth gene of $\mathbf{G}$ and $\mathbf{G}^*$, respectively. Each time missing values are randomly introduced to the dataset.

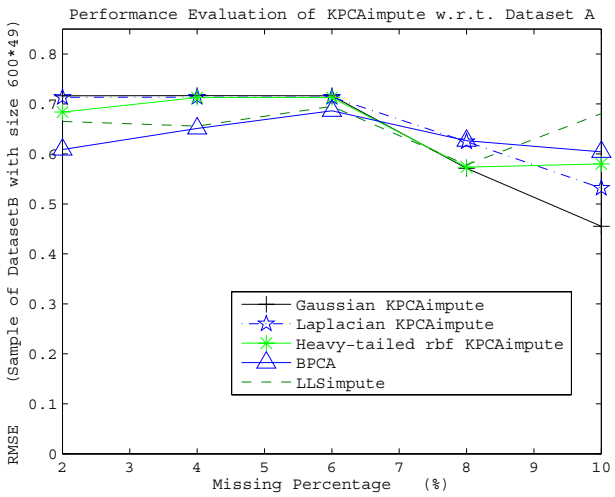

Fig. 1. Performance of KPCAimpute w.r.t. Dataset A

B. The Kernel Functions

We consider the heavy-tailed kernels with elements defined as

$$\mathbf{K}_{ij} = \exp(-\rho\sum_{n=1}^{N}||[\mathbf{g}_i(n)]^a - [\mathbf{g}_j(n)]^a||^p), \quad (24)$$

where $\mathbf{g}_i(n)$ and $\mathbf{g}_j(n)$ are the nth elements of the ith and jth genes in the observation matrix $\mathbf{G}$ $(i, j = 1 \cdots M)$. The parameter a can be simply understood as a remapping of the input, but not in terms of kernel products. As illustrated in [10], a does not affect the dimension but may improve robustness. We use the SVM kernel methods toolbox [11] to generate kernel functions. The parameter ρ is set to 1 in the software by the renormalization of the input data, and $a \leq 1$ and $p \leq 2$ according to [11]. In particular, we use a Laplacian kernel [11], where $a = 0.5$ and $p = 1$, and a heavy-tailed radial basis function (rbf) kernel [11], where $a = 0.5$ and $p = 2$. It is apparent that when $a = 1$ and $p = 2$, the kernel is a Gaussian kernel.

C. Performance Evaluation

In the experiments, we sample the two microarrays in sizes of 600 genes. We run the algorithms 20 times, where for each run the data is sampled from a different part of the original datasets. The RMSE result for each algorithm is therefore the average. As mentioned above, three kernel functions are evaluated for KPCAimpute, including heavy-tailed rbf kernel, Laplacian kernel and Gaussian kernel. The performaces of the three KPCAimpute are compared with those of the BPCA and LLSimpute. We set the number of principal axes $L = M - 1$ for KPCAimpute and BPCA imputation, if we do not have a prior knowledge on the datasets, as used in [4]. Fig. 1 shows the performance evaluation of the algorithms with respect to Dataset A, while Fig. 2 shows that with respect to Dataset B.

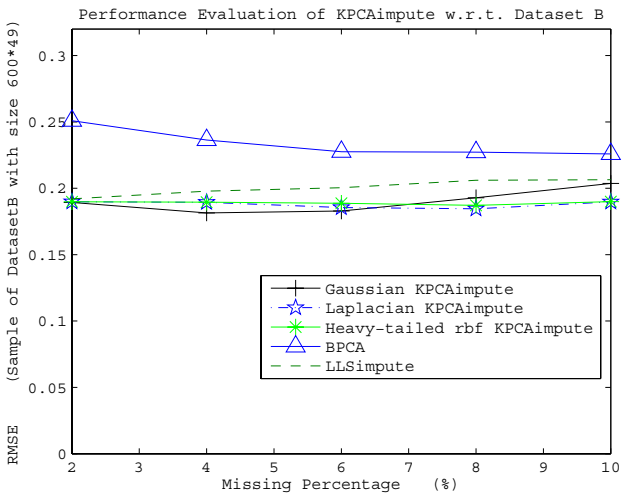

Fig. 2. Performance of KPCAimpute w.r.t. Dataset B

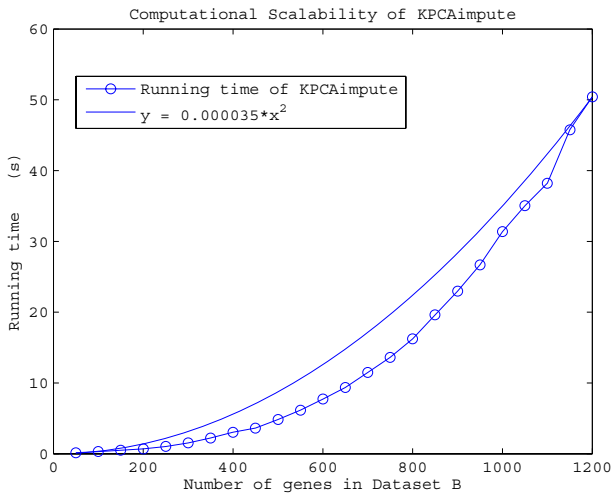

Fig. 3. Computational Scalability of KPCAimpute Algorithm

It can be seen from Fig. 1 that the KPCAimpute outperforms the LLSimpute when the missing percentage increases (greater than 8%, say). And the KPCAimpute shows comparative performance to that of the BPCA. Among the three kernel functions, the Gaussian KPCAimpute performs best for Dataset A, comparing to the other two kernel PCA imputations. In Fig. 2, we see that the KPCAimpute algorithms can outperform both the BPCA and LLSimpute. For Dataset B, the Laplacian kernel and Gaussian kernel are both promising kernel functions to use.

In TABLE I, we present the standard deviations (STD) of the RMSE results of different imputation algorithms when 2% missing values are applied to Dataset B. It can be seen that the KPCAimpute algorithms outperform the BPCA and are similar to the LLSimpute in terms of prediction stability, but avoiding the heuristic computation procedure of the LLSimpute.

TABLE I
THE STANDARD DEVIATIONS OF THE RMSE RESULTS FOR DATASET B

	Gauss	Laplace	Heavy-tailed	BPCA	LLSimpute
RMSE	0.1893	0.1899	0.1899	0.2509	0.1920
STD	0.0061	0.0061	0.0062	0.0749	0.0060

In addition, the computational scalability of the KPCAimpute was investigated. Fig. 3 shows the computation time versus the number of genes in Dataset B, where Gaussian kernel is used and 2% missing values are applied in the algorithm. The computational cost can be approximated by a slow increasing curve in quadratic-like form.

IV. Summary

In this paper, we study the kernel PCA based imputation algorithms for microarray data analysis. It is a generalization of the standard PCA imputation. The advantage of the KPCAimpute is that it explores the features of the observed data and uses them as regressors in estimating missing values. The KPCAimpute outperforms the LLSimpute when the missing percentage increases. The performance of the KPCAimpute is similar to or better than that of BPCA. The Gaussian kernel shows effective performance in the KPCAimpute. In summary, KPCAimpute is a promising method and a good alternative to BPCA in microarray missing data estimation.

References

[1] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, pp. 467–470, 1995.

[2] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirina, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[3] H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: local least squares imputation," *Bioinformatics*, vol. 21, no. 2, pp. 187–198, 2005.

[4] S. Oba, M. A. Sato, I. Takemasa, M. Monden, K. I. Matsubara, and S. Ishii, "A Bayesian missing value estimation method for gene expression profile data," *Bioinformatics*, vol. 19, no. 16, pp. 2088–2096, 2003.

[5] B. Schölkopf and A. J. Smola, *Learning with kernels*. USA: The MIT Press, 2002.

[6] Z. Liu, D. Chen, and H. Bensmail, "Gene expression data classification with kernel principal component analysis," *Journal of Biomedicine and Biotechnology*, vol. 2, pp. 155–159, Jun. 2005.

[7] Z. Liu, D. Chen, H. Bensmail, and Y. Xu, "Clustering gene expression data with kernel principal component analysis," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 2, pp. 303–316, Apr. 2005.

[8] D. J. C. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[9] R. Rosipal, "Kernel-based regression and objective nonlinear measures to assess brain functioning," Ph.D. dissertation, University of Paisley, Scotland, Sept. 2001.

[10] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[11] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and kernel methods Matlab toolbox," Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.