2013 IEEE 13th International Conference on Data Mining

Noise-Resistant Bicluster Recognition

Huan Sun
CS Department
University of California
Santa Barbara, USA
huansun@cs.ucsb.edu

Gengxin Miao
ECE Department
University of California
Santa Barbara, USA
miao@umail.ucsb.edu

Xifeng Yan
CS Department
University of California
Santa Barbara, USA
xyan@cs.ucsb.edu

***Abstract*—Biclustering is crucial in finding co-expressed genes and their associated conditions in gene expression data. While various biclustering algorithms (e.g., combinatorial, probabilistic modelling, and matrix factorization) have been proposed and constantly improved in the past decade, data noise and bicluster overlaps make biclustering a still challenging task. It becomes difficult to further improve biclustering performance, without resorting to a new approach. Inspired by the recent progress in unsupervised feature learning using deep neural networks [1], in this work, we propose a novel model for biclustering, named AutoDecoder (AD), by relating biclusters to features and leveraging a neural network that is able to automatically learn features from the input data. To suppress severe noise present in gene expression data, we introduce a non-uniform signal recovery mechanism: Instead of reconstructing the whole input data to capture the bicluster patterns, AD weighs the zero and non-zero parts of the input data differently and is more flexible in dealing with different types of noise. AD is also properly regularized to deal with bicluster overlaps. To the best of our knowledge, this is the first biclustering algorithm that leverages neural network techniques to recover overlapped biclusters hidden in noisy gene expression data. We compared our approach with four state-of-the-art biclustering algorithms on both synthetic and real datasets. On three out of the four real datasets, AD significantly outperforms the other approaches. On controlled synthetic datasets, AD performs the best when noise level is beyond 15%.**

Source Code: http://grafia.cs.ucsb.edu/autodecoder/

***Keywords*-Gene Expression, Biclustering, Neural Network**

I. Introduction

High-throughput gene expression profiling is readily accessible as the development of new technologies such as the Affymetrix array plates and next-generation sequencing, which necessiates advanced analysis tools to deal with massive amounts of data. To analyze expression data, numerous computational tools have been developed and steadily improved, among which, biclustering becomes a dominant unsupervised technique for gene expression analysis.

Biclustering refers to a process of grouping genes and conditions simultaneously, producing a set of biclusters each including a gene set and a condition set. Fig. 1 shows two overlapped biclusters in a sample dataset. The gene expression values "1", "-1" and "0" indicate upregulating, downregulating, and unchanged, respectively. As shown in the figure, several important characteristics exist in bicluster recognition:

(1) There are various kinds of biclusters: Genes (conditions) can be positively and negatively correlated;

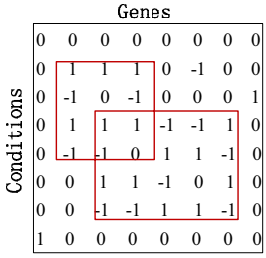

Fig. 1. Biclusters in a Sample Dataset.

(2) Biclusters can overlap with each other in both gene dimension and condition dimension since multiple pathways containing the same gene could be active under different conditions;

(3) It is not necessary that each gene (or condition) has to participate in at least one bicluster (not necessarily full coverage);

(4) Bicluster detection shall be robust against heavy noise in the input data.

Due to these unique characteristics, many non-overlap, full-coverage clustering methods are ineffective to recognize embedded biclusters.

To tackle the aforementioned challenges, many biclustering algorithms have been developed. We can roughly classify them into three categories: combinatorial methods, e.g., CTWC [2], OPSM [3], ISA [4], BIMAX [5], association analysis based RAP [6], COALESCE [7], and QUBIC [8]; probabilistic and generative approaches such as SAMBA [9], FABIA [10] and many others [11], [12], [13]; matrix factorization methods like spectral clustering [14], SSVD [15], and S4VD [16]. While biclustering performance has been significantly improved in the past decade, data noise and bicluster overlaps make the problem still quite challenging. Our controlled experiments will show that when noise level or bicluster overlap degree is high, most of existing algorithms can only discover a small percentage of biclusters. In this work, we are going to re-examine the biclustering problem from a new perspective and demonstrate the superior performance of our proposed model.

Inspired by the recent progress in unsupervised feature learning through deep neural networks (i.e., deep learning) [1], we relate the biclustering problem to feature learning where

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.34

the expression pattern of the gene set under the condition set in a bicluster can be regarded as a feature. Different variants of deep neural networks, such as [17], [18], have recently been tested to have tremendous power in extracting high-level features in images. A deep neural network is composed of multiple hidden layers, where a higher (lower) hidden layer extracts features at a higher (lower) level. In our biclustering setting, we relate the expression of one gene under all conditions to the pixel values of one image. Our interested expression pattern corresponds to the correlations among pixels, which are essentially low-level features. Therefore, we save ourselves the trouble of building a *deep* neural network; instead, we resort to one neural network with a single hidden layer. Nevertheless, we expect the unsupervised feature learning power of neural networks will still boost the biclustering performance.

Particularly, we generalize sparse autoencoder (SAE), a recently proposed neural network for unsupervised feature learning [19], to mine biclusters. While SAE focuses on reconstructing the whole data, we are less interested in data reconstruction, than in extracting bicluster patterns. This motivates us to develop a model that can selectively reconstruct the data by putting different penalties on reconstruction errors in bicluster and non-bicluster areas in the input. This design significantly improves the performance over SAE and makes it possible to outperform the existing highly optimized biclustering algorithms. We name this model *AutoDecoder* (AD).

Our method is also related to matrix factorization, but significantly differs in the following aspects. First, matrix factorization is a linear method while the neural network in our model incorporates a non-linear activation layer. PCA or SVD can be considered as a special case when the hidden layer picks a linear activation function. Second, we do not search for a low-dimensional approximation to the whole expression data; instead, we allow partial false reconstructions based on the location of noise, inside or outside biclusters. Third, a sparse activation constraint is imposed on the hidden neurons, which allows us to artificially control the number of genes to be discovered in each bicluster. With these advantages, AD is demonstrated to be more accuate than the leading matrix factorization approaches such as SSVD [15] and S4VD [16].

To the best of our knowledge, this is the first work to relate the biclustering problem to feature learning and apply neural network approaches to biclustering gene expression data. Our motivation is based on the conjecture that since biclustering has been a long-standing topic, we might need to tackle the problem from a new perspective that has never been explored by previous studies in detail. Our model parameters are learnt using back propagation and a second order optimization algorithm L-BFGS [21], which have been popularly exploited in the deep learning literature. According to the optimized model parameters, we then design a bicluster embedding strategy to determine the genes and conditions in a bicluster. Empirically, our biclustering method runs very fast and outperforms the most up-to-date leading methods such as QUBIC, COALESCE, FABIA, and S4VD.

The rest of the paper is organized as follows. We first briefly introduce sparse autoencoder, the predecessor of our to-be-proposed model, and make a connection between neural networks and biclustering in Section II. Our method, AutoDecoder, is presented in Section III, with discussions on noise resistance analysis and bicluster patterns. Experimental results on both real and synthetic datasets are given in Section V. Related works are reviewed in Section VI. Finally, we discuss future work and conclude in Section VII.

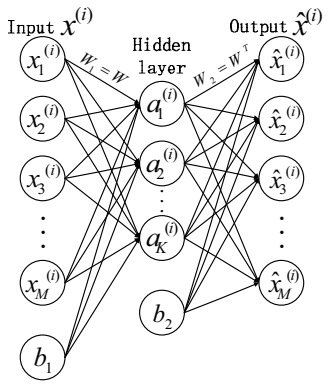

Fig. 2. Graphical Representation of an Autoencoder.

II. Preliminaries

In this section, we build a connection between biclustering and the sparse autoencoder (SAE) [19]. Then we discuss the necessity to develop a generalized model AutoDecoder .

The graphical representation of an autoencoder with a single hidden layer is shown in Fig. 2. We begin with clarifying the notations in the context of biclusering.

(1) X: an $M \times N$ gene expression matrix, where M is the number of conditions and N is the number of genes. $x_m^{(i)}$ denotes the expression value of the i_{th} gene under the m_{th} condition.

(2) W: a $K \times M$ matrix, the combination weights between the input layer and the hidden layer. Each element $w_{k,m}$ represents the contribution of the expression under the m_{th} condition to bicluster k, and determines whether the m_{th} condition is a member in bicluster k or not. The weights in the first layer and those in the second layer are optionally tied together by W_1=W_2^T=W [1]. b_1: a $K \times 1$ vector; b_2: an $M \times 1$ vector. b_1 and b_2 work as the bias terms at the input layer and the hidden layer respectively.

(3) $a^{(i)}$: a $K \times 1$ vector representing activation values of the i_{th} gene in K hidden neurons, where each of the K hidden neurons denotes a potential bicluster. Unless otherwise stated, we employ the popular element-wise sigmoid activation function, i.e., $a^{(i)} = \text{sigmoid}(Wx^{(i)} + b_1)$. Each component $a_k^{(i)} \in (0, 1)$ encodes how likely a gene belongs to bicluster k. Let $A = [a^{(1)}, a^{(2)}, \ldots, a^{(i)}, \ldots, a^{(N)}]$ be the activation matrix of all the genes.

(4) $\hat{X}$: reconstructed gene expression data. $\hat{X} = [\hat{x}^{(1)}, \hat{x}^{(2)}, \ldots, \hat{x}^{(i)}, \ldots, \hat{x}^{(N)}]$, where $\hat{x}^{(i)} = W^T a^{(i)} + b_2$.

Parameters in an autoencoder are learnt by minimizing the reconstruction error between X and $\hat{X}$.

For each hidden neuron, its activation rate is defined as $\hat{\rho}_k = \sum_{i=1}^{N} a_k^{(i)}/N$. Let a $K \times 1$ vector ρ be our expected activation rates of the K hidden neurons. The Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ_k and a Bernoulli random variable with mean $\hat{\rho}_k$ is added to the objective function as an additional penalty term. Autoencoder with this sparsity penalty term is called sparse autoencoder (SAE) [19]. This term can be entitled with more meanings in our setting. In the biclustering context, $\hat{\rho}N$ can be regarded as the number of genes each bicluster actually contains after learning, which is regularized to be close to ρN we *expect* each bicluster to contain. Usually, it is a desideratum to obtain a bicluster which is neither too small nor too large. By using this term, the number of genes in one bicluster can be controlled artificially, which cannot be achieved by most of the existing biclustering algorithms.

SAE learns the optimal model parameters by minimizing the cost $\frac{1}{2N}\|X - \hat{X}\|_F^2 + \beta \sum_{k=1}^{K} \mathrm{KL}(\rho_k \| \hat{\rho}_k) + \frac{\lambda}{2}\|W\|_F^2$, where $\|\cdot\|_F$ is the Frobenius norm. SAE can work well when the expression data are uniformly corrupted by the same Gaussian noise (the same mean and variance). However, in practice, Gaussian random noise inside and outside biclusters could be different. In such situation, putting equal weights on reconstructing the bicluster part and non-bicluster part of the expression data will hurt bicluster finding. Therefore, we develop a non-uniform weighting strategy to discriminatively treat the background and the patterns. We call the new model, *AutoDecoder* (AD), to emphasize that it decodes the biclusters rather than encodes the entire expression data.

III. Framework

In this section, we introduce our model AutoDecoder (AD), followed by discussions on model solution and bicluster recognition processes.

A. Optimization Formulation

AutoDecoder shares the same neural network structure as shown in Fig. 2. However, it enhances sparse autoencoder (SAE) to be more robust against noise and bicluster overlaps.

$$\begin{aligned}\arg\min_{W,b_1,b_2} \mathcal{H} \quad &= \underbrace{\frac{1}{2N}\sum_{i=1}^{N}\sum_{m=1}^{M}[I_{m,i} + \alpha(1 - I_{m,i})](\hat{x}_m^{(i)} - x_m^{(i)})^2}_{\text{(I)}} \\ &+ \underbrace{\beta \sum_{k=1}^{K} \mathrm{KL}(\rho_k \| \hat{\rho}_k)}_{\text{(II)}} \\ &+ \underbrace{\frac{\gamma}{2}\sum_{k=1}^{K}\sum_{i\neq j} S_{i,j}(|W_{k,i}| - |W_{k,j}|)^2}_{\text{(III)}} \\ &+ \underbrace{\lambda \|W\|_1}_{\text{(IV)}}\end{aligned}$$

The above objective function is a sum of four terms, where β, γ, and λ take non-negative values and control the trade-off among different regularization terms. We now explain each of the terms in details:

- Term (I) quantifies the average reconstruction error with non-uniform weights. I is an indicator matrix with $I_{m,i} = 0$ denoting $x_m^{(i)} = 0$ and $I_{m,i} = 1$ denoting $x_m^{(i)} \neq 0$. α controls the relative weight of reconstructing the zero values and non-zero values in the input data. One can check that when $\alpha = 1$, AD will put the same emphasis on reconstructing the zero values and non-zero values (uniform weighting). In this case, this term degenerates to the reconstruction function in SAE. Conceptually, our model is suitable for any data with background as zeros and patterns as non-zeros. In this work, we focus on the discretized matrix with elements {-1, 0, 1}, representing down-regulating, unchanged, and up-regulating.
- Term (II) is the sparsity penalty term as in SAE. To make the activation of each hidden neuron as sparse as we desire, we constrain the KL-divergence between the sparsity parameter ρ_k and the activation rate $\hat{\rho}_k$: $\mathrm{KL}(\rho_k \| \hat{\rho}_k) = \sum_{k=1}^{K} \rho_k \log \frac{\rho_k}{\hat{\rho}_k} + (1 - \rho_k) \log \frac{1-\rho_k}{1-\hat{\rho}_k}$.
- In Term (III), S is an $M \times M$ symmetric matrix, where $S_{i,j}$ is the absolute value of cosine similarity between the i_{th} condition (row) and j_{th} condition in matrix X. If $S_{i,j}$ is large enough, two conditions i and j should have similar membership after optimization ($|W_{k,i}| \sim |W_{k,j}|$, for any k). This term prevents dramatic membership change of two similar conditions, which will enhance the robustness against bicluster overlaps.
- Term (IV) is a regularizer named as weight decay (Chapter 5 in [20]), where $\|W\|_1$ is the L_1 norm of W, i.e., $\|W\|_1 = \sum_{k=1}^{K}\sum_{m=1}^{M}|W_{k,m}|$. L_1 norm regularization tends to recover W with more sparsity by forcing the insignificant values to zeros, which will make easier the bicluster membership interpretation. We neglect the bias terms b_1 and b_2 because their inclusion will make the results of W depend on the choice of origin (Section 1.1 and 5.5.1 in [20]).

B. Model Solution

The classic L-BFGS algorithm [21], [22] is applied to minimize the objective function $\mathcal{H}$. L-BFGS needs the derivatives of parameters. They are calculated by the classic backpropagation algorithm [23], where error messages are split at each neuron and then propagated to the neurons at previous layers. Given the current parameters ($W^{K\times M}, b_1^{K\times 1}, b_2^{M\times 1}$), the backpropagation procedure is performed as follows:

1. Perform a forward pass, i.e, compute the activation matrix A of the hidden layer and the reconstructed matrix $\hat{X}$ of the output layer.

2. Let $\delta^{(1)} \in R^{K\times N}$ and $\delta^{(2)} \in R^{M\times N}$ denote the error terms of the hidden layer and the output layer, respectively. Compute the error terms $\delta^{(2)}$ and $\delta^{(1)}$ as follows:

$$\delta_{m,n}^{(2)} = [I_{m,n} + \alpha(1 - I_{m,n})](\hat{x}_m^{(n)} - x_m^{(n)})$$

$$\delta_{k,n}^{(1)} = [\sum_{m=1}^{M} W_{k,m}\delta_{m,n}^{(2)} + \beta(-\frac{\rho_k}{\hat{\rho}_k} + \frac{1-\rho_k}{1-\hat{\rho}_k})]\, a_k^{(n)}(1 - a_k^{(n)})$$

3. Compute the derivatives of $W^{K\times M}, b_1^{K\times 1}, b_2^{M\times 1}$:

$$\frac{\partial \mathcal{H}}{\partial b_{1\,k}} = \frac{1}{N}\sum_{n=1}^{N}\delta_{k,n}^{(1)}$$

$$\frac{\partial \mathcal{H}}{\partial b_{2\,m}} = \frac{1}{N}\sum_{n=1}^{N}\delta_{m,n}^{(2)}$$

$$\frac{\partial \mathcal{H}}{\partial W_{k,m}} = \frac{1}{N}\sum_{n=1}^{N}(\delta_{k,n}^{(1)}x_m^{(n)} + \delta_{m,n}^{(2)}a_k^{(n)}) + (\gamma\, L_{m,(:)}W_{k,(:)}^T + \lambda)\,\mathrm{sgn}(W_{k,m})$$

where L is an $M \times M$ symmetric matrix with $L_{i,j} = -S_{i,j}$ when $i \neq j$, and $L_{i,i} = \sum_{1\leq j\leq M, j\neq i} S_{i,j}$. $L_{m,(:)}$, $W_{k,(:)}$ are the m_{th} and k_{th} row of L and W respectively. $\mathrm{sgn}(\cdot)$ is the sign function of a real number and takes 0 as the subgradient of the absolute value function $y = |x|$ at $x = 0$.

Once the objective function $\mathcal{H}$ and its derivatives are computed, the L-BFGS algorithm is applied to minimize $\mathcal{H}$. During optimization, instead of storing the dense Hessian approximation matrix, the L-BFGS algorithm saves only a few vectors to represent the approximation implicitly, which significantly decreases the memory requirement. Since L-BFGS is a classic and well-established algorithm [21], [22], we omit the detailed analysis on its complexity and convergence rate. In practice, we observe that the L-BFGS algorithm converges very fast (within around 1000 iterations on all the real datasets).

C. Bicluster Recognition

In order to detect the embedded biclusters, we need to select both genes and conditions for each bicluster. The weight $|W_{k,m}|$ determines the contribution of the m_{th} condition to the k_{th} bicluster. If $|W_{k,m}|$ is large enough, the bicluster should contain the m_{th} condition. Similarly for selecting genes into one bicluster, if the activation value $a_k^{(i)}$ is high, the i_{th} gene should belong to bicluster k.

The bicluster embedding process is summarized as follows:
For each hidden neuron k, i.e., each potential bicluster,

- Gene selection
  Pick any gene i corresponding to $|a_k^{(i)}| > \delta$ ($\delta \in (0, 1)$);
- Condition selection
  Pick any condition m corresponding to $|W_{k,m}| > \epsilon$ ($\epsilon \in (0, 1)$).

The thresholds δ and ϵ can be tuned by users; their values are fixed at 0.7 and 0.3 respectively for all of our experiments unless otherwise stated. Intuitively, the same threshold should work for both the gene and condition selection; however, elements in weight matrix W are usually much smaller due to the weight decay regularizer; we pick different thresholds respectively for the gene and condition selection.

D. Robustness Against Noise

One of the major contributions we made is the non-uniform weighting of the zero values and non-zero values in the input data. As shown in Fig. 1, biclusters contain expression values representing upregulating ($+1$) or downregulating (-1).

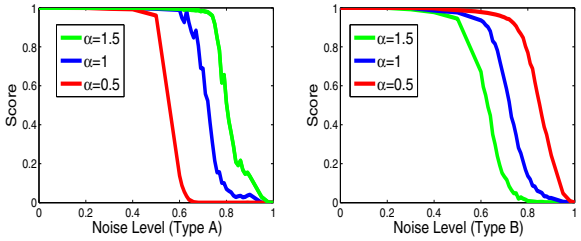

Fig. 3. Performance of AD with $\alpha = \{0.5, 1, 1.5\}$ w.r.t Different Noise Types.

Therefore, there are two kinds of noise: Type A noise corresponding to the cases where nonzeros appear outside the biclusters, and Type B noise corresponding to the cases where zeros appear inside a bicluster. Accordingly, there are two kinds of reconstruction errors: (1) false negative error when $+1$ or -1 is reconstructed as 0; (2) false positive error when 0 is reconstructed as $+1$ or -1. According to Term (I) in $\mathcal{H}$, penalty weight on the false positive error is controlled by α while penalty weight on the false negative error is set to 1. When α is greater than 1, AD allows more false negative errors in the reconstruction process. In this situation, AD tends to exclude ± 1's out from the final patterns than to include 0's into the patterns, which is robust against Type A noise. In real gene expression datasets, we first run existing algorithms to detect biclusters and estimate the dominant noise type. In the case where Type A noise occurs more often than Type B noise, $\alpha > 1$ is recommended.

Here we verify our analysis on AD's robustness against noise. We generate a set of matrices of size 100×500. In each matrix, 100 of the 500 columns are 1's while the others are 0's. The area in the matrix containing 1's is called pattern area while the area containing 0's is called background area. We consider two simple, yet illustrative, scenarios. In the first scenario, the elements in background area are flipped to 1's with probability p ranging from 0 to 1, whereas the pattern area is kept clean (Type A noise). The flipping probability p is named *noise level*. Oppositely, in the second scenario, the elements in pattern area are flipped to 0's with probability p (noise level) ranging from 0 to 1 whereas the background area is kept clean (Type B noise). At each noise level, we generate 50 matrices and compute the average F score (See Section V-A). We show the result of AD with $\alpha = 0.5, 1, 1.5$ in Fig. 3. $\alpha = 1$ corresponds to the uniform weighting as in SAE. Obviously, with an appropriate α our proposed model is more robust than the classic SAE approach.

E. Robustness Against Bicluster Overlaps

Many of existing biclustering algorithms cannot detect biclusters accurately when biclusters overlap at both gene and condition dimensions. Term (III) in $\mathcal{H}$ is designed to deal with such two-dimension overlaps. Our intuition is that if two conditions (genes) have similar expression values, they should have similar bicluster membership. That is, for a condition

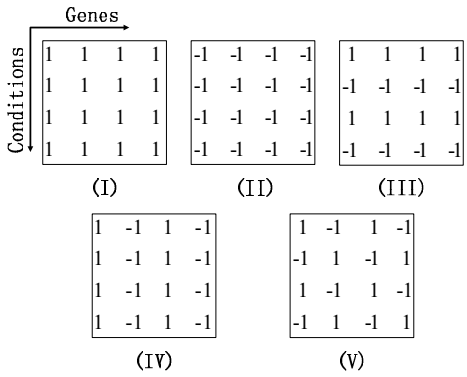

Fig. 4. Bicluster Patterns that Can Be Discovered by AD.

pair (i, j) and any bicluster k, $(|W_{k,i}| - |W_{k,j}|)^2$ should be as small as possible if the gene expression values in condition i is similar to that in condition j. Furthermore, if one condition is similar to conditions in multiple biclusters, it should belong to these biclusters simultaneously, thus resulting in biclusters overlapped in conditions. Formally, $\sum_{k=1}^{K} S_{i,j}^{(condition)}(|W_{k,i}| - |W_{k,j}|)^2$ should be minimized where $S_{i,j}^{(condition)}$ is the similarity between the expression values under condition i and condition j. Similarly for a gene pair (i, j), $\sum_{k=1}^{K}\sum_{i\neq j} S_{i,j}^{(gene)}(a_k^{(i)} - a_k^{(j)})^2$ should be minimized where $S_{i,j}^{(gene)}$ is the similarity between the expression values of gene i and gene j across all the conditions. Due to our neural network structure $a^{(i)} = \text{sigmoid}(Wx^{(i)} + b_1)$, if two genes have similar expression values ($x^{(i)} \sim x^{(j)}$), they will naturally have similar activation values ($a^{(i)} \sim a^{(j)}$). Therefore, in AD, only the condition term is included, and $S^{(condition)}$ is replaced with S for simplicity.

IV. Bicluster Patterns

AutoDecoder can identify a broad range of bicluster patterns in one trial. Fig. 4 shows five types of bicluster patterns that can be found by AD.

(I-III) In these three patterns, conditions can be either positively or negatively correlated under the gene set whereas genes are *positively* correlated under the condition set. Our AD model can directly discover such patterns. For some k, the weights $W_{k,m}$ corresponding to the condition set will be large in magnitude. The sign of the $W_{k,m}$ will be the same as that of the expressions under the m_{th} condition. The gene set will have large activation values in the k_{th} neuron.

(IV-V) In the fourth and fifth patterns, conditions can be either positively or negatively correlated under the gene set whereas genes can also be either *positively* or *negatively* correlated under the condition set. For such patterns, our AD model with a sigmoid hidden layer splits the bicluster into two subclusters in which the two subsets of genes behave oppositely. This is because column 1 (3) and column 2 (4) cannot simultaneously have large activation values under the same hidden neuron. Since both subclusters are associated with

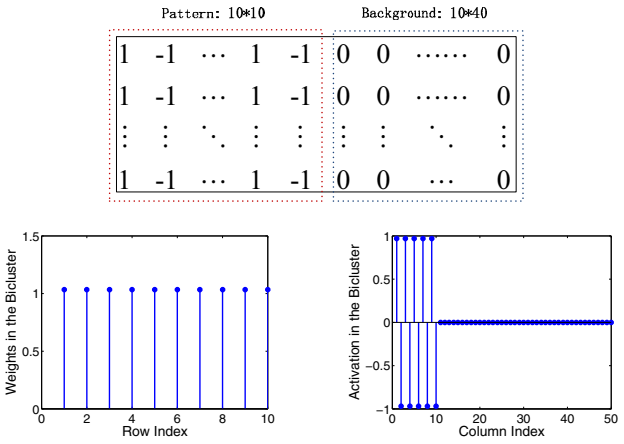

Fig. 5. Bicluster Recognition with Pattern IV.

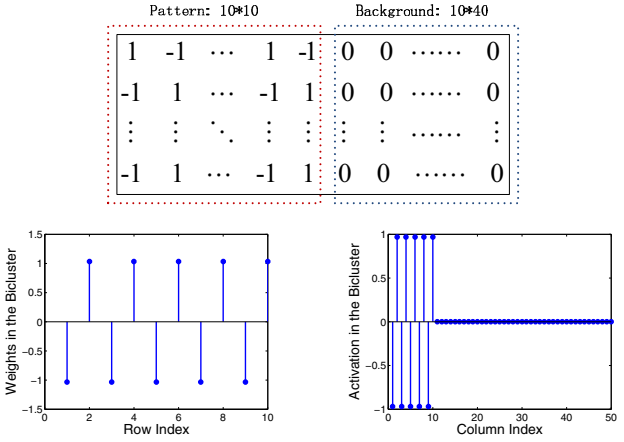

Fig. 6. Bicluster Recognition with Pattern V.

the same condition set in our embedding process, they can be combined through postprocessing. Alternatively, if the sigmoid activation function in the hidden layer is replaced by a tanh activation function, these patterns can be directly discovered.

We are going to show that AD with a tanh activation layer is able to discover Patterns IV and V. Recall that in Section II, $a^{(i)} = \text{sigmoid}(Wx^{(i)} + b_1)$. Now we replace sigmoid with a tanh function, i.e., $a^{(i)} = \tanh(Wx^{(i)} + b_1) \in (-1, 1)$. As long as $|a^{(i)}|$ is large, we regard the i_{th} gene to be active. In this case, the original definition of the sparsity parameter $\hat{\rho}_k = \sum_{i=1}^{N} a_k^{(i)}/N$ is problematic; therefore, we re-define $\hat{\rho}_k = \sum_{i=1}^{N} {a_k^{(i)}}^2/N$. One can also use $\hat{\rho}_k = \sum_{i=1}^{N} |a_k^{(i)}|/N$, however, we favor the former due to its smoothness. All other components in our original AD model are kept unchanged.

We generate two toy matrices of size 10×50, in which the 10 rows and the first 10 columns form a bicluster with Pattern IV and Pattern V respectively. The number of biclusters K is set at 1. Fig. 5 and 6 show the membership of rows and

columns for a bicluster with Pattern IV and V respectively. The upper subfigure shows a simulated matrix with Pattern IV or V. The lower left subfigure provides the learnt weights of the 10 rows ($W_{1,m} : m = 1, \ldots, 10$) while the lower right subfigure illustrates the activation values of the 50 columns ($A_{1,n} : n = 1, \ldots, 50$). According to the bicluster recognition process discussed in Section III-C, both the columns and rows belonging to the bicluster can be successfully identified due to a high magnitude in either weights or activation values.

V. Experiments

In this section, we present empirical studies of AutoDecoder (AD) on both real gene expression datasets and controlled synthetic datasets, and compare AD with the state-of-the-art biclustering algorithms in different categories: (1) Combinatorial search, Qubic [8] and COALESCE[7], (2) Probabilistic model, FABIA [10], and (3) Matrix factorization, S4VD [16]. These four algorithms represent the best results in their own category after a decade of efforts in searching high-quality biclusters in gene expression data. A detailed comparison of many other existing algorithms with FABIA is available in [10].

AD was implemented in MATLAB. All the experiments were conducted on a 3.4GHZ, 16GB, Intel PC running Windows 7.

A. Evaluation Measures

To assess various biclustering approaches, we use the average *Relevance* and *Recovery* scores, which were widely adopted in biclustering literature, such as S4VD [16], QUBIC [8], and BIMAX [5]. Overall, these two scores quantify the similarity between the set of computed biclusters and the set of true biclusters in the data.

Suppose M^* is the set of true biclusters, and M is the set of discovered biclusters each of which contains a gene set G and a condition set C. We denote M^* and M respectively as $M^* = \{M_1^*, M_2^*, ..., M_{m^*}^*\}$ and $M = \{M_1, M_2, ..., M_m\}$. Let $M_i^* = G_i^* \times C_i^*$ and $M_i = G_i \times C_i$, where $\times$ is the Cartesian product. The average *Relevance* and *Recovery* scores are defined as follows:

$$\textit{Relevance} = \frac{1}{m}\sum_{i=1}^{m} \max_{j\in\{1,2,..,m^*\}} \frac{M_i \cap M_j^*}{M_i \cup M_j^*}$$

$$\textit{Recovery} = \frac{1}{m^*}\sum_{i=1}^{m^*} \max_{j\in\{1,2,..,m\}} \frac{M_i^* \cap M_j}{M_i^* \cup M_j}$$

where $\frac{M_i \cap M_j^*}{M_i \cup M_j^*}$ is the well-known Jaccard coefficient [24]. The average *Relevance* evaluates to what degree the discovered biclusters represent the true biclusters, whereas the average *Recovery* measures how well the true biclusters are recovered by the current biclustering algorithm. We calculate the harmonic mean of the *Relevance* and *Recovery* scores, $F = \frac{2Rel.*Rec.}{Rel.+Rec.}$, in order to jointly consider them. Note that although there are various kinds of evaluation measures available for general clustering problems, for biclustering, we select the most widely adopted *Relevance* and *Recovery* measures in the literature.

The performance on all the simulated datasets is evaluated using F score computed as above. However, for real gene expression datasets, ground truth on the bicluster membership of genes is usually not available, therefore we can only compute the scores on the condition clusters each of which corresponds to the condition set within a bicluster, i.e., $M_i^* = C_i^*$ and $M_i = C_i$. Meanwhile, the biological significance of the gene set within a detected bicluster is evaluated. This strategy was also adopted in our competing algorithms [10], [16].

B. Gene Expression Datasets

We experimented four real gene expression datasets widely used in biclustering studies: breast cancer, multiple tissue types, diffuse large-B-cell lymphoma, and lung cancer. For the first three datasets, Hoshida *et al.* [25] clustered the conditions using additional datasets and verified the clusters by gene set enrichment analysis. FABIA [10] studied how well biclustering algorithms can re-identify these clusters without any additional information. Similarly, for the lung cancer dataset, S4VD [16] and SSVD [15] first compared the obtained condition clusters with ground truth and then evaluated gene sets within biclusters by doing enrichment analysis. We adopted a similar approach to evaluate the detected biclusters. We briefly describe the datasets as follows:

(a) Breast cancer dataset [26]. As stated in [10], the sample (i.e., condition) S54 is an outlier and was removed from the original data. The dataset has 1,213 genes and 97 samples.

(b) Multiple tissue types dataset [27]. It contains 5,565 genes and 102 samples from diverse tissues and cell lines. Hoshida *et al.* [25] discovered four sample clusters from this dataset.

(c) Diffuse large-B-cell lymphoma (DLBCL) dataset [28]. This dataset was to predict the survival after chemotherapy, and contains 3,795 genes and 58 samples. Four sample clusters were discovered in [25].

(d) Lung cancer dataset [29]. It was previously analyzed in [16], [15]. 12,625 genes under 56 samples were measured using the Affymetrix 95av2 GeneChip.

1) Preprocessing: For those gene expression datasets that are not valued among $\{-1, 0, 1\}$, we adopted a discretization procedure similar to QUBIC [8]. Specifically, for each gene, sort its original expression values on M conditions in the increasing order as follows:$(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(s-1)}, x_{\pi(s)}, \ldots, x_{\pi(c-1)}, x_{\pi(c)}, x_{\pi(c+1)}, \ldots, x_{\pi(M-s+1)}, x_{\pi(M-s+2)}, \ldots, x_{\pi(M)})^T$, where $\pi(j)$ is the permutated j_{th} index, $c = \lceil M/2 \rceil$ and $s - 1 = \lfloor Mq \rfloor$. Same as [8], we set q at 0.06. A condition is considered as downregulating (-1) for a gene if its expression value is less than $x_{\pi(c)} - d$ and as upregulating $(+1)$ if it is greater than $x_{\pi(c)} + d$, where $d = \min\{x_{\pi(c)} - x_{\pi(s)}, x_{\pi(M-s+1)} - x_{\pi(c)}\}$; otherwise, the gene is not activated (i.e., 0). The rationale to choose this procedure is given in the Supplementary Data of [8]. The choice is not exclusive; other discretization methods can

also be applied here. Since we find that FABIA, S4VD, and COALESCE do not work better on preprocessed data, the raw data are used as their input.

2) Parameter Setting: Biclustering algorithms generally involve many parameters. We explored the number of biclusters in all the methods except COALESCE from 5 to 8, slightly larger than the true number in each dataset, as suggested in FABIA. COALESCE can find any number of biclusters that satisfy their criteria. In these popularly used real datasets, only a small number of biclusters are present; nevertheless, we will later synthesize datasets with more biclusters embedded for comprehensive comparison. For each competing algorithm, we tuned 2 to 4 important parameters according to their manual to show their best performance on each dataset.

For AD, we observe that a fixed setting of parameters generalize very well across different datasets. Particularly, we set $\beta = \frac{2M}{\sqrt{N}}$, $\gamma = \frac{0.2}{N}$, and $\lambda = \frac{0.1}{\sqrt{N}}$. Two important parameters in AD are α and ρ. There are three typical values for α: 1.5, 0.5, and 1, which can deal with Type A noise, Type B noise, and balanced Type A and B noise as discussed in Section III-D. For ρ, it will be easier to set if one has any prior knowledge or preference over the number of genes contained in biclusters. In our experiments, we first run other algorithms to get a rough idea about the noise type and gene cluster size. In the real gene expression datasets we tested, Type A noise occurs more often than Type B noise, and therefore, we set α at 1.5. We randomly draw each component of ρ from the interval $[0.01, 0.03]$, as we expect a bicluster to contain $1\% \sim 3\%$ the total number of genes.

3) Performance Comparisons: On all of these datasets, AD completes biclustering within 2 minutes. The results are summarized in Fig. 7. Columns "$Rel.$", "$Rec.$", and "F" respectively provide the average *Relevance* score, average *Recovery* score and their harmonic mean. The columns "#g", "#s", and "#bc" give the average number of genes, conditions in each bicluster, and the number of discovered biclusters respectively. "TN" gives the true number of condition clusters in each dataset. For the breast cancer dataset, FABIA performs the best. For the remaining three datasets, AD significantly outperforms other tuned biclustering algorithms. Understanding the differences between the breast cancer dataset and other datasets will be helpful to further improve the performance, which we leave for future study. The superiority of AD over SAE verifies the effectiveness of our proposed strategy in dealing with real data. Moreover, AD can achieve better performance once the parameters are tuned as in our competing algorithms.

To evaluate the biological significance of the gene set in a discovered bicluster, as in [8], [10], [16], we conducted gene ontology (GO) enrichment analysis by computing the P-value [8] of each biological category *w.r.t* the gene set. Particularly, we caclulate the probability of having r ($r > 0$) genes of the same biological category in a bicluster with n genes as:

$$\Pr(r|N, m, n) = \frac{\binom{m}{r}\binom{N-m}{n-r}}{\binom{N}{n}} \quad (1)$$

Methods	Breast Cancer (TN: 3)					
	$Rel.$	$Rec.$	F	#g	#s	#bc
AD	0.44	0.42	0.43	15	23	3
SAE	0.47	0.48	0.48	20	30	6
QUBIC	0.42	0.14	0.21	33	15	1
COALESCE	0.34	0.52	0.41	79	36	11
FABIA	0.67	0.49	**0.57**	98	31	4
S4VD	0.60	0.41	0.49	124	27	2
Methods	Multiple Tissue (TN: 4)					
	$Rel.$	$Rec.$	F	#g	#s	#bc
AD	0.75	0.89	**0.82**	290	23	5
SAE	0.63	0.70	0.66	270	34	5
QUBIC	0.58	0.67	0.63	103	18	5
COALESCE	0.52	0.44	0.47	546	77	3
FABIA	0.77	0.77	0.77	98	31	4
S4VD	0.19	0.07	0.10	245	8	1
Methods	DLBCL (TN: 4)					
	$Rel.$	$Rec.$	F	#g	#s	#bc
AD	0.48	0.53	**0.50**	103	17	5
SAE	0.48	0.48	0.48	118	24	5
QUBIC	0.34	0.35	0.34	174	7	6
COALESCE	0.36	0.39	0.38	289	40	6
FABIA	0.55	0.18	0.27	99	17	1
S4VD	0.34	0.24	0.28	270	18	2
Methods	Lung Cancer (TN: 4)					
	$Rel.$	$Rec.$	F	#g	#s	#bc
AD	0.89	0.96	**0.92**	442	14	5
SAE	0.60	0.76	0.67	476	15	6
QUBIC	0.65	0.59	0.62	130	12	4
COALESCE	0.29	0.48	0.36	167	14	89
FABIA	0.85	0.85	0.85	607	13	4
S4VD	0.78	0.68	0.72	445	16	3

Fig. 7. Results on Real Datasets.

where N is the total number of genes in the input, and m is the number of genes from that biological category and encoded in the input. For each biological category with $r > 0$ in one bicluster, we calculate its P-value *w.r.t* the gene set in the bicluster using the probability defined by (1). We then use the smallest P-value among all possible biological categories as the P-value of the current bicluster. The smaller the P-value, the more biologically significant the bicluster [8], [16], [10]. Our model can generally discover biclusters with P-value around or less than 10^{-4}, much often less than 10^{-10} on all the datasets, which is comparable to the gene set analysis result shown by FABIA [10] and S4VD [16]. Due to space limit, we only show the summarized gene enrichment analysis results on the last three datasets.

For Multiple Tissue dataset, Bicluster 1 is enriched with genes related to the acyl-CoA metabolic process and the thioester metabolic process (P-value 2.8×10^{-6}). Genes related to collagen fibril organization enrich Bicluster 2 (P-value 4.0×10^{-6}). The most significant GO terms in Bicluster 3 are related to immune response (P-value 1.2×10^{-20}). Genes

in Bicluster 4 are related to positive regulation of superoxide anion generation (P-value 6.9×10^{-5}). The most significant GO terms in Bicluster 5 are related to G-protein coupled receptor protein signaling pathway (P-value 6.9×10^{-7}).

For DLBCL dataset, genes in Bicluster 1 are most related to the regulation of transcription involved in G1/S phase of mitotic cell cycle (P-value 1.1×10^{-4}). Bicluster 2 is highly related to defense response (P-value 6.3×10^{-10}). Bicluster 3 is enriched by genes from the regulation of peptide hormone secretion. The most significant GO terms in Bicluster 5 are highly related to the regulation of phospholipase activity (P-value 5.3×10^{-4}). Bicluster 4 is too small to allow for a reliable biological interpretation.

In the discovered biclusters from Lung Cancer dataset, Bicluster 1 is enriched with genes related to positive regulation of kidney development (P-value 2.2×10^{-6}). The most significant GO terms in Bicluster 2 are highly related to cell division (P-value 9.2×10^{-26}). Bicluster 3 is related to interferon-gamma-mediated signaling pathway (P-value 1.2×10^{-12}). Genes related to mitochondrial electron transport enrich Bicluster 4 (P-value 3.9×10^{-8}). The most significant biological category in Bicluster 5 is related to cell adhesion (P-value 2.2×10^{-10}).

The evaluation of biclustering results illustrates that our approach not only improves the performance on condition clusters of these datasets, but also guarantees the biological significance of the gene sets discovered in the biclusters.

4) Sensitivity Analysis: For most of the learning problems, model selection is a critical problem. The learning performance might vary significantly under different parameter settings. β, γ, and λ are our model parameters that control the trade-off among the regularization terms. We fix $\beta = \frac{2M}{\sqrt{N}}$, $\gamma = \frac{0.2}{N}$, and $\lambda = \frac{0.1}{\sqrt{N}}$ on the previous experiments. In this subsection, we study the impact of the parameters on the performance of AD. Specifically, we fix other parameters as before and let one of $\{\beta, \gamma, \lambda\}$ vary. As shown in Fig. 8, AD is generally not very sensitive with respect to these parameters, since high quality scores can be achieved under a large range of the studied parameters.

C. Synthetic Datasets

Our algorithm is also tested on a set of controlled synthetic datasets, which are generated as follows: Given the matrix size 100×500 and the number of biclusters K, the number of rows r in a bicluster is randomly selected from the interval $[10, 30]$, and the number of columns c is randomly selected from $[50, 100]$; then we randomly choose r rows from the total 100 rows and c columns from the total 500 columns as the members of the bicluster. The matrix is initially filled with "0". Each bicluster is filled with "1". In total, K biclusters are generated. We then inject noise to each matrix by flipping the value of elements. Specifically, we flip the 1's inside biclusters to be zeros with probability p and flip the 0's outside biclusters to be 1 or -1 respectively with probability $\frac{p}{2}$. The flipping probability p is named *noise level*.

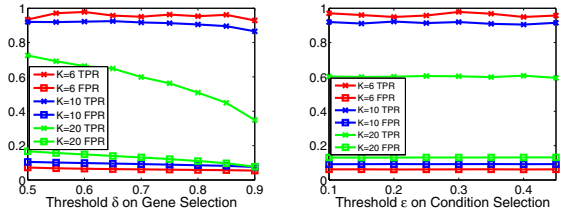

Fig. 10. True Positive Rate (TPR) and False Positive Rate (FPR) of AD.

For each parameter setting K and p, 100 matrices are generated and the average performance of each algorithm on these 100 matrices is measured. We tune parameters in our competing algorithms and show their best performance. For AD, we still use a fixed parameter setting, where $\alpha = 1$ and $\rho_k = 0.1, \forall k$.

Fig. 9 illustrates the performance of five algorithms. It is observed that AD significantly outperforms all the other algorithms when noise level is beyond 15%. When we increase the number of biclusters, recognition becomes more and more difficult because overlapping among biclusters also increases significantly. For example, around 12% of the nonzeros belong to two biclusters when $K = 10$, whereas the ratio increases to 22% when $K = 20$. In such situation, all the competitors are only able to detect a much smaller percentage of true biclusters, while AD still maintains a much higher F score. Fig. 9 shows that AD covers the situation (high noise and high overlaps) where the existing algorithms cannot work well.

We further show the change of the true positive rate (TPR) and false positive rate (FPR) of AD, when varying the recognition thresholds δ and ϵ in Section III-C. We detail how to caculate the TPR and FPR in our biclustering setting: We first map each detected bicluster M_i to a true bicluster M_j^* by maximizing the Jaccard coefficient, i.e, by $\max_j \frac{M_i \cap M_j^*}{M_i \cup M_j^*}$. We compute the TPR and FPR for each true bicluster, and then obtain their average as the overall TPR and FPR of AD. The above synthetic datasets with noise level at 0.2 are used for study. Fig. 10 shows at 0.2 noise level, AD generally still achieves a high TPR while keeping the FPR at a low level. On the other hand, AD turns out to be not very sensitive to the thresholds, since similar results can be achieved under a large range of the thresholds.

VI. Related Work

The first biclustering algorithm for gene expression analysis was proposed in the year 2000 by Cheng and Church [30]. Since then, various combinatorial biclustering algorithms have been developed, including the Coupled Two-Way Clustering [2], the Order-Preserving SubMatrix [3], the Iterative Signature Algorithm [4], BIMAX developed by [5], QUBIC [8], and COALESCE [7]. An association rule based method RAP [6] also falls into this category. We include the recent QUBIC and COALESCE for comparative studies in our experiments. Computational complexity and performance have

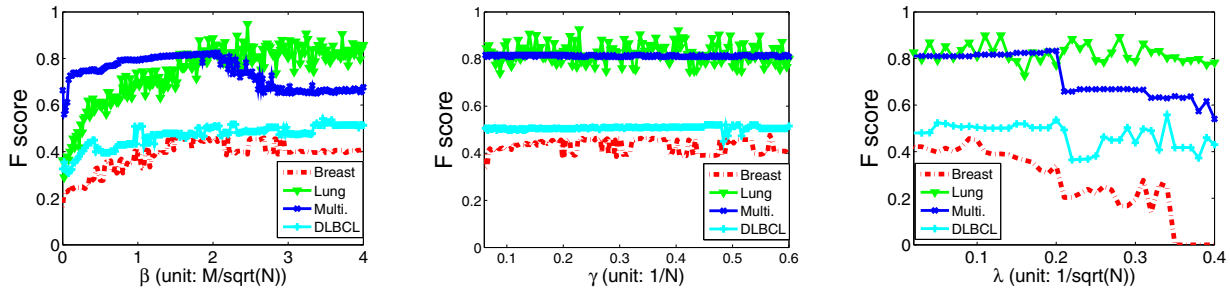

Fig. 8. Sensitivity Analysis of $\{\beta, \gamma, \lambda\}$ on Real Datasets.

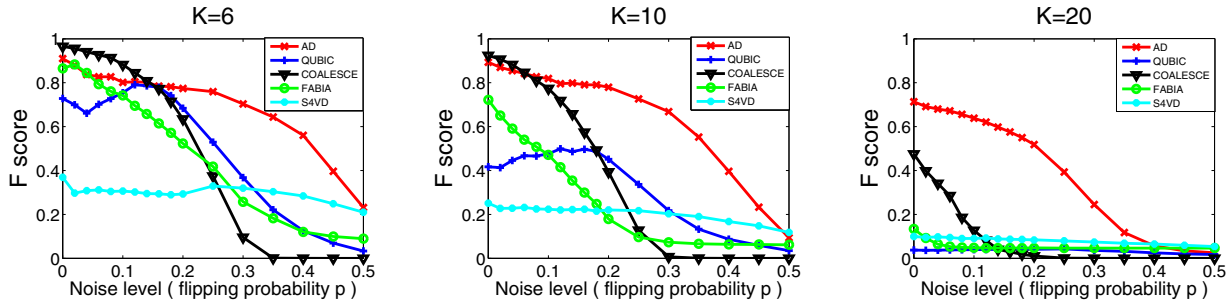

Fig. 9. Results on Synthetic Datasets

been steadily improved as the development of new algorithms. Probabilistic and generative methods are also available for biclustering gene expression data. The statistical-algorithmic method for bicluster analysis (SAMBA [9]) is one of the earliest probabilistic approaches to biclustering. Other approaches include those proposed in [11], [12], [13]. FABIA [10] is one of the recently developed probabilistic approaches, and has demonstrated good performance on various real datasets. Another category of biclustering algorithms is based on matrix factorization. Spectral biclustering [14] applies singular vector decomposition (SVD) to extracting expression patterns in a matrix. It is indicated in [31] that SVD is capable of finding biclusters. Lee *et al.* [15] developed a sparse SVD (SSVD) algorithm. S4VD [16] improved SSVD [15] by incorporating a stability selection technique. Recently, query-based biclustering algorithms [32], [33] are developed, which utilize a set of seed genes provided by the user to prune the search space and guide the biclustering algorithm. Hanczar *et al.* [34] focus on the corrected measurement of the biclustering methods. Despite of many existing studies on biclustering, we are motivated to further improve the performance by approaching the problem from a novel perspective.

Cross association [35], co-clustering [36], [37], [38], [39], exclusive row biclustering [40], monochromatic biclustering proposed in [41], block models [42], Boolean Matrix Factorization [43], [44], and techniques on noisy information theoretic tiles detection [45] have been applied to group rows and columns simultaneously in a matrix. Each of these methods also falls into one of the three previously mentioned categories, i.e., combinatorial approach, probabilistic modelling, or matrix factorization. These existing models usually have one or more of the following characteristics: (1) Only 0-1 binary input can be handled, which is impossible to detect negative correlations; (2) Disjoint/non-overlapping or full-coverage biclusters are generated, which is different from our emphasis introduced in Section I; (3) They rely on stochastic approximation to do inference due to lack of computationally efficient algorithms.

Our work is inspired by deep learning, i.e., unsupervised feature learning using deep neural networks. Since the breakthrough on training multi-layer neural networks was made by Hinton and Salakhutdinov[1], numerous studies on learning deep feature hierarchies have been conducted. Among them, the most related one is sparse autoencoder proposed in [19]. Different from sparse autoencoder, our model is more resistant against noise by differently reconstructing the bicluster and background part of the input data. Besides, we also enhance the ability of sparse autoencoder to deal with bicluster overlaps. Through extensive experiments, we clearly demonstrate that our neural network based approach can outperform the existing biclustering algorithms, as well as the original sparse autoencoder.

VII. Conclusion and Future Work

We developed a novel biclustering approach, AutoDecoder, to effectively discover biclusters in highly noisy expression data. To the best of our knowledge, this is the first attempt to relate the biclustering problem to unsupervised feature learning methods and apply neural network approaches to biclustering gene expression data. AutoDecoder associates the activation of hidden neurons in a two-layer neural network with the membership of genes and conditions in biclusters. Compared with four state-of-the-art algorithms, AutoDecoder

performs better on both real and synthetic datasets, especially when there are more overlapped biclusters and higher noise. Our experimental results show that neural network is a promising approach to biclustering, a long-standing problem in gene expression data analysis.

Apart from evaluation in terms of *Relevance* and *Recovery* scores, and P-value, it will be interesting to verify whether the biclusters recognized by AutoDecoder are more useful to biologists in the future. Biclustering methods usually involve many parameters. Another very useful future work will be to test different parameter settings to fully explore their potential, especially for AutoDecoder. Since biclustering has been a long-standing topic, to further improve the performance, one might need to explore new ideas different from existing biclustering methodologies.

ACKNOWLEDGMENT

This work was partially supported by the U.S. National Science Foundation under grant IIS-0954125 and the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

[1] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, p. 504, 2006.

[2] G. Getz, E. Levine, and E. Domany, "Coupled two-way clustering analysis of gene microarray data," *PNAS*, vol. 97, no. 22, p. 12079, 2000.

[3] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving submatrix problem," *J. of Computational Biology*, vol. 10, no. 3-4, pp. 373–384, 2003.

[4] J. Ihmels, S. Bergmann, and N. Barkai, "Defining transcription modules using large-scale gene expression data," *Bioinformatics*, vol. 20, no. 13, pp. 1993–2003, 2004.

[5] A. Prelić, S. Bleuler, P. Zimmermann *et al.*, "A systematic comparison and evaluation of biclustering methods for gene expression data," *Bioinformatics*, vol. 22, no. 9, pp. 1122–1129, 2006.

[6] G. Pandey, G. Atluri, M. Steinbach, C. L. Myers, and V. Kumar, "An association analysis approach to biclustering," in *SIGKDD*. ACM, 2009, pp. 677–686.

[7] C. Huttenhower, K. Mutungu, N. Indik, W. Yang, M. Schroeder, J. Forman, O. Troyanskaya, and H. Coller, "Detailing regulatory networks through large scale data integration," *Bioinformatics*, vol. 25, no. 24, pp. 3267–3274, 2009.

[8] G. Li, Q. Ma, H. Tang, A. Paterson, and Y. Xu, "Qubic: a qualitative biclustering algorithm for analyses of gene expression data," *Nucleic acids res*, vol. 37, no. 15, pp. e101–e101, 2009.

[9] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, no. suppl 1, pp. S136–S144, 2002.

[10] S. Hochreiter, U. Bodenhofer, M. Heusel *et al.*, "Fabia: factor analysis for bicluster acquisition," *Bioinformatics*, vol. 26, no. 12, pp. 1520–1527, 2010.

[11] Q. Sheng, Y. Moreau, and B. De Moor, "Biclustering microarray data by gibbs sampling," *Bioinformatics*, vol. 19, no. suppl 2, p. ii196, 2003.

[12] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," *Statistica Sinica*, vol. 12, no. 1, pp. 61–86, 2002.

[13] J. Gu and J. Liu, "Bayesian biclustering of gene expression data," *BMC genomics*, vol. 9, no. Suppl 1, p. S4, 2008.

[14] Y. Kluger, R. Basri, J. Chang, and M. Gerstein, "Spectral biclustering of microarray data: coclustering genes and conditions," *Genome research*, vol. 13, no. 4, pp. 703–716, 2003.

[15] M. Lee, H. Shen, J. Huang, and J. S. Marron, "Biclustering via sparse singular value decomposition," in *Biometrics*, vol. 66, 2010, pp. 1087–1095.

[16] M. Sill, S. Kaiser, A. Benner, and A. Kopp-Schneider, "Robust biclustering by sparse singular value decomposition incorporating stability selection," *Bioinformatics*, vol. 27, no. 15, pp. 2089–2097, 2011.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*. ACM, 2009, pp. 609–616.

[18] Q. V. Le, R. M. MarcAurelio Ranzato, K. C. Matthieu Devin, and J. D. Greg Corrado, "Building high-level features using large scale unsupervised learning," in *ICML*. ACM, 2012, pp. 131–138.

[19] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," *NIPS*, vol. 20, pp. 873–880, 2008.

[20] C. Bishop, *Pattern recognition and machine learning*, 2006.

[21] D. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[22] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer Science+ Business Media, 2006.

[23] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *IEEE Int. Conf. on Neural Networks*, vol. 1. IEEE, 1996, pp. 347–352.

[24] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 107–145, 2001.

[25] Y. Hoshida, J. Brunet, P. Tamayo, T. Golub, and J. Mesirov, "Subclass mapping: identifying common subtypes in independent disease data sets," *PLoS ONE*, vol. 2, 2007.

[26] L. van't Veer, H. Dai, V. D. Vijver *et al.*, "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, vol. 415, no. 6871, pp. 530–536, 2002.

[27] A. Su, M. Cooke, K. Ching *et al.*, "Large-scale analysis of the human and mouse transcriptomes," *PNAS*, vol. 99, no. 7, p. 4465, 2002.

[28] A. Rosenwald, G. R. Wright, W. Chan *et al.*, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma," *New England Journal of Medicine*, vol. 346, no. 25, pp. 1937–1947, 2002.

[29] A. Bhattacharjee, W. Richards, J. Staunton *et al.*, "Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses," *PNAS*, vol. 98, no. 24, p. 13790, 2001.

[30] Y. Cheng and G. Church, "Biclustering of expression data," in *ISMB*, vol. 8, 2000, pp. 93–103.

[31] S. Busygin, O. Prokopyev, and P. Pardalos, "Biclustering in data mining," *Computers & Operations Research*, vol. 35, no. 9, pp. 2964–2987, 2008.

[32] H. Zhao, L. Cloots *et al.*, "Query-based biclustering of gene expression data using probabilistic relational models," *BMC Bioinformatics*, vol. 12, 2011.

[33] F. Alqadah, J. S. Bader, R. Anand, and C. K. Reddy, "Query-based biclustering using formal concept analysis."

[34] B. Hanczar and M. Nadif, "Precision-recall space to correct external indices for biclustering," in *ICML*, 2013, pp. 136–144.

[35] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos, "Fully automatic cross-associations," in *SIGKDD*, 2004, pp. 79–88.

[36] I. Dhillon, S. Mallela, and D. Modha, "Information-theoretic co-clustering," in *SIGKDD*. ACM, 2003, pp. 89–98.

[37] H. Shan and A. Banerjee, "Bayesian co-clustering," in *ICDM*. IEEE, 2008, pp. 530–539.

[38] M. Shafiei and E. Milios, "Latent dirichlet co-clustering," in *ICDM*. IEEE, 2006, pp. 542–551.

[39] G. Bisson and C. Grimal, "Co-clustering of multi-view datasets: a parallelizable approach," in *ICDM*. IEEE, 2012, pp. 828–833.

[40] A. Painsky and S. Rosset, "Exclusive row biclustering for gene expression using a combinatorial auction approach," in *ICDM*. IEEE, 2012, pp. 1056–1061.

[41] S. Wullf, R. Urner, and S. Ben-David, "Monochromatic bi-clustering," in *ICML*, 2013.

[42] E. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed membership stochastic blockmodels," *JMLR*, vol. 9, pp. 1981–2014, 2008.

[43] Z. Zhang, T. Li, C. Ding, X. Ren, and X. Zhang, "Binary matrix factorization for analyzing gene expression data," *Data Min Knowl Disc*, vol. 20, pp. 28–52, 2010.

[44] P. Miettinen and J. Vreeken, "Model order selection for boolean matrix factorization," in *SIGKDD*. ACM, 2011, pp. 51–59.

[45] K. Kontonasios and T. De Bie, "An information-theoretic approach to finding informative noisy tiles in binary databases," 2010.