BIOINFORMATICS

Vol. 00 no. 00 2008
Pages 1–8

An Unsupervised Conditional Random Fields Approach for Clustering Gene Expression Time Series

Chang-Tsun Li *, Yinyin Yuan and Roland Wilson *

Department of Computer Science, University of Warwick, Coventry, UK

Associate Editor:

ABSTRACT

Motivation: There is a growing interest in extracting statistical patterns from gene expression time series data, in which a key challenge is the development of stable and accurate probabilistic models. Currently popular models, however, would be computationally prohibitive unless some independence assumptions are made to describe large scale data. We propose an unsupervised conditional random fields model to overcome this problem by progressively infusing information into the labelling process through a samll variable voting pool.

Results: An unsupervised conditional random fields model (CRF) is proposed for efficient analysis of gene expression time series and is successfully applied to gene class discovery and class prediction. The proposed model treats each time series as a random field and assigns an optimal cluster label to each time series, so as to partition the time series into clusters without a priori knowledge about the number of clusters and the initial centroids. Another advantage of the proposed method is the relaxation of independence assumptions.

Availability:

Contact: ctli@dcs.warwick.ac.uk

1 BACKGROUND

A key challenge in genomic signal research is the development of efficient and reliable probabilistic models for analysing gene expression data. As gene expression is a temporal process, it is necessary to record a time course of gene expression in order to determine the set of genes that are expressed under certain conditions, the gene expression levels and the interactions between these genes. While static data are assumed to be time-independent, time course data have strong correlations between successive points. This allows us to fully utilise the information from the experiments, as it reveals the pathway that leads from one state to the next, instead of a stable state, under given conditions.

Nevertheless, as time course data are obtained from multiple microarrays produced at different time points with replicates, the large amount of data causes difficulties for analysis. This requires appropriate models to identify patterns in such large scale datasets. To this end, gene expression clustering provides an efficient way to discover groups in large scale datasets. The underlying assumption in clustering gene expression data is that co-expression indicates co-regulation or relation in functional pathways, so an efficient clustering method should identify genes that share similar functions (Boutros and Okey, 2005). Quite often, the ground truth about the number of clusters in the dataset and the cluster memberships of the genes are unknown, making classifier training infeasible. As such, unsupervised clustering methods requiring no training and no *a priori* knowledge about the number of clusters are desirable.

Many gene clustering methods for gene class discovery have been proposed in the literature (Culotta *et al.*, 2005; Heard *et al.*, 2006; Husmeier, 2003; Schliep *et al.*, 2005; Wu *et al.*, 2005). One research focus is the development of stable and accurate probabilistic models. The advantage of probabilistic methods over some alternatives is that they allow measurements of uncertainty (Culotta *et al.*, 2005; Heard *et al.*, 2006; Husmeier, 2003; Luan and Li, 2003; Ng *et al.*, 2006). Generative probabilistic models, such as Dynamic Bayesian Networks (DBN's), are trained to maximise the joint probability of a set of observed data and their corresponding labels (Dojer *et al.*, 2006; Husmeier, 2003). Very often, model generality comes at the cost of computational inefficiency. For instance, mixture models work well for static experiments. However, when it comes to time series clustering, there are $k \times n^2$ parameters to be estimated for the k covariance matrices, n being the number of time points (Medvedovic *et al.*, 2004). In addition, mixture models are not effective for time course data, as they treat time points as discrete and ignore their time order (Ma *et al.*, 2006). Hidden Markov Models (HMM's), a special case of DBN's, are another popular method for probabilistic sequence data modelling and have been applied in the fields such as speech recognition and language sequence modelling. As a relatively straightforward machine learning method, they have also been successfully applied to gene expression time course clustering (Ji *et al.*, 2003; Schliep *et al.*, 2005). A first-order HMM model can be expressed as a joint probability distribution of the observed data x and the corresponding label / state y, such that

$$p(x, y) = \prod_i p(x_i|y_i)p(y_i|y_{i-1}) \quad (1)$$

This model is also referred to as a generative model. The term $p(y_i|y_{i-1})$ reflects the Markov property that the probability of a future state y_i, given past states, depends only on current state y_{i-1}, while the term $p(x_i|y_i)$ can be viewed as a likelihood, or observation on x_i. In order to be computationally tractable, they assume that each data is generated independently by a hidden process. However, this is not always the case in many applications (Lafferty *et al.*, 2001). Furthermore, in gene expression data modelling, it is impossible to represent gene interactions between genes for generative models because enumerating all possibilities is intractable.

*to whom correspondence should be addressed

Conditional probabilistic models such as Conditional Random Fields (CRF's) (Culotta *et al.*, 2005; Lafferty *et al.*, 2001), on the other hand, define a conditional probability over labels given observations. They label a pattern / time series by selecting the label sequence that maximises the conditional probability, without relying on independence assumptions. They are efficient and typically outperform generative probabilistic models such as HMM's in a number of tasks such as language sequence modelling and gene sequence prediction (Culotta *et al.*, 2005; Lafferty *et al.*, 2001) because they allow arbitrary dependencies on observations. Let X be a random variable over the observations and Y be a random variable over the corresponding labels. A CRF model can be expressed as a joint distribution of Y given X

$$\begin{aligned} p(Y|X) &= p(X|Y)p(X) \\ &\propto \exp(-U(X,Y,\Lambda)) \end{aligned} \tag{2}$$

where U is a cost function, which guides the search for the optimal clustering. Albeit its advantages as mentioned above, the main limitation of CRF's is the low convergence rate if the whole dataset is involved (Lafferty *et al.*, 2001). Therefore, a proper model should have a structure that can facilitate faster convergence, which, in our proposed system, is achieved by inferring labels conditioned on the information acquired from a neighbourhood system, called *voting pool*, as described in the next section.

2 METHODS

This section explains how the gene expression time series are transformed into a multi-dimensional feature space, with the temporal correlation and time intervals taken into account, and how the proposed CRF model is formulated. In order not to confuse the reader with details, we first present the whole picture of the proposed algorithm as follows.

Algorithm 1 Conditional Random Fields Clustering Algorithm

1. Transform each τ-time-point time series into a $\tau - 1$ dimensional feature vector.
2. Assign a random label to each time series to form an initial label configuration.
3. For each time series i
   3.1 Form a voting pool, N_i.
   3.2 Calculate the cost functions of the labels of the members in the voting pool.
   3.3 Calculate the probabilities of the labels based on their cost functions.
   3.4 Assign the label with the greatest probability y_j to time series i if its current label y_i is not equal to y_j.
4. Go back to Step 3 if the label of any time series is updated, otherwise stop

2.1 Data transformation

Given a set of τ-time-point data, to take the temporal relationship and interval between time points into account, a simple linear transformation is carried out to transform the dataset into a $\tau - 1$ dimensional feature space. Let n denote the number of time series, i.e., the number of genes. Each original time series T_i of length τ, $i = 1, 2, ..., n$ is now mapped to a point x_i in the $\tau - 1$ dimensional feature space with the tth component, $x_i(t)$, defined as

$$x_i(t) = \frac{T_i(t+1) - T_i(t)}{I_t}, \qquad t = 1, 2, 3, \ldots, \tau - 1 \tag{3}$$

where I_t is the time interval between t and $t + 1$. The numerator thus designed in Eq.(3) is to capture causality and the temporal dependency of gene expression time series while the involvement of the denominator is to reflect the fact that the length of intervals between time points is important information.

2.2 Model formulation

As mentioned in Section 1, involving the whole dataset incurs high computational complexity and low convergence rate. To alleviate this problem, we exploit the local characteristic (also known as *Markovianity*) of Markov random fields to condition the learning process of our model. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of n observed time series and $Y = \{y_1, y_2, \ldots, y_n\}$ the set of the corresponding labels. The objective of the learning is to assign an optimal class label y_i to time series i conditioned on observed data x_i, observed data x_j and class labels y_j, for all j in a small neighbourhood N_i, which we call the *voting pool* (See Section 2.2.1 for details). Therefore, instead of taking the general form of Eq. (2), the proposed model can be specifically expressed as

$$P(y_i|x_i, x_{N_i}, y_{N_i}, \Lambda) = P(x_i, x_{N_i}|y_i, y_{N_i}, \Lambda)P(y_i|y_{N_i}, \Lambda) \tag{4}$$

where $P(x_i, x_{N_i}|y_i, y_{N_i}, \Lambda)$ and $P(y_i|y_{N_i}, \Lambda)$ are the conditional probability distribution and the prior, respectively, and Λ is a set of parameters, as described later. For the sake of conciseness, we will use x_{N_i} and y_{N_i} to represent the observed data and class labels of all the members in N_i, respectively. We will also drop Λ from the writing of the model.

The conditional random field model of Eq. (4) can also be expressed in the Gibbs form (Geman and Geman, 1984) in terms of cost functions $U_i^c(x_i, x_{N_i}|y_i, y_{N_i})$ and $U_i^p(y_i|y_{N_i})$, which are associated with the conditional probability and the prior of Eq.(4), respectively, as

$$P(y_i|x_i, x_{N_i}, y_{N_i}) \propto \exp[-(U_i^c(x_i, x_{N_i}|y_i, y_{N_i}) + U_i^p(y_i|y_{N_i}))] \tag{5}$$

As also mentioned in the previous section that unsupervised clustering methods requiring no training and no *a priori* knowledge about the number of clusters are desirable, the design of the proposed clustering algorithm is based on the postulates that these two requirements can be met by

- Not using cluster centroids in the labelling process, but the relative similarity between each time series and the member time series in a randomly formed voting pool. By random we mean, for each time series, the members in its voting pool are different in different iterations.
- Allowing each individual time series to be a singleton cluster and to interact with the time series in the voting pool to find its own identity progressively.

Before the first iteration of the labelling process starts, each time series is assigned a label randomly picked from the integer range $[1, n]$, where n is the number of samples. Therefore, the algorithm starts with a set of n singleton clusters without the user specifying the number of clusters.

2.2.1 Voting pool In each iteration, when a time series i is visited, its voting pool N_i is formed with k time series. The voting pool includes s *most-similar (MS)* time series in Euclidean space, one *most-different (MD)* time series and $k-s-1$ time series selected at random. The most-similar and the most-different time series are selected from the voting pool once the learning process starts. That is, in any iteration, if a voting time series, j, selected at random is more similar to time series i than the sth most-similar member is to i, the sth most-similar member is replaced by j. By the same token, if a voting time series, j, selected at random is more different from time series i than the current most-different member is from i, the current most-different member is replaced by j. Subsequently, each time series is assigned an optimal label decided by a cost function defined later. The most-different (MD) member plays the role of informing the algorithm of what label (i.e., the one associated with the MD member) to avoid while the most-similar (MS) members are for maintaining local information and informing the algorithm what labels to choose. The $k-s-1$ members selected at random are for introducing new information, which includes global and local information depending on what new members (intra-class or inter-class) are selected, in the learning process, and for replacing the MD and MS members, if more appropriate one are selected. Therefore, we can see that, without involving the shear number of time series of the whole dataset, as the algorithm iterates, the local information becomes more accurate and more information are infused into the voting process. It is this progressive manner of information infusion that reduces the computational complexity.

To allow the algorithm to work without the user specifying the number of clusters, when each time series is visited, only the class labels currently assigned to the members of its voting pool are taken as candidates. The advantages of randomising the initial label configuration with a label space of n labels and using the proposed voting pool are now clear. First, globally the number of labels allowed is equal to the number of time series, n. Secondly, the computational complexity is kept low because locally the optimal label of each individual time series is sampled from a small label space with its cardinality less than k because there are k members in the voting pool and some may have the same labels. The cardinality of this small label space will decrease in due course as the homogeneous clusters progressively form. Although starting from a random initial label configuration, through the interaction with the members of the voting pool, a unique optimal label for each cluster can be quickly identified.

The voting pool size k, as one of the parameter in Λ of Eq. (4), is determined by a few factors such as the size of dataset and desired computing efficiency. The impact of k is analysed in Section 3.1 and tabulated in Table 1.

The reason clusters merge and eventually converge to a certain number of final clusters is twofold. First, the s most similar time series of each time series in question, i, encountered since the initial iteration are always kept in i's voting pool and the algorithm tends to assign the label possessed by the majority of the s similar members according to the cost function defined in the next subsection. Secondly, at each iteration, $k-s-1$ random time series are selected to be new members of i's voting pool and any of the s most similar members in the voting pool may be replaced if more similar time series are among the new members. These new members allow diversity to be introduced into the voting pool. It is the *majority* of the s most similar members and the *diversity* brought in by the new members that facilitate cluster merge and eventually converge to a certain number of final clusters.

2.2.2 Cost function Like the objective functions in other optimisation problems, the two cost functions $U_i^c(\cdot)$ and $U_i^p(\cdot)$, of Eq. (5) encodes the observed data and labels of the members of the voting pool, and aims to encourage "good labelling" with low cost and penalise "poor labelling" with high cost. Defining a feasible cost function is important, but by no means trivial. First, since the two cost functions in Eq. (5) are dependent on the same set of arguments, by combining the two, we obtain a new model as

$$P(y_i|x_i, x_{N_i}, y_{N_i}) \propto \exp(-U_i(y_i, y_{N_i}, x_i, x_{N_i})) \tag{6}$$

Secondly, as the proposed algorithm does not require the user to specify the number of clusters and requires no information about cluster centroids, therefore instead of using the distances between time series and cluster centroids, the observed data is encoded in the form of pair-wise potential, $W_{i,j}$, between paired time series i and j, defined as

$$W_{i,j}(y_i, y_j, x_i, x_j) = \begin{cases} -(D - d_{i,j}) & \text{if } y_i = y_j \\ (D - d_{i,j}) & \text{if } y_i \neq y_j \end{cases} \tag{7}$$

where $d_{i,j}$ is the Euclidean distance between time series i and j. D is the estimated threshold dividing the set of Euclidean distances into intra- and inter-class distances. To estimate D, for each time series, we calculate its distances to m other randomly picked time series and find the minimum and maximum distances. Then two values, d_p and d_o, are calculated by taking the average of all the minimum and maximum distances, respectively. Finally, D is defined as

$$D = \frac{1}{2}(d_p + d_o) \tag{8}$$

m is another parameter of Λ in Eq. (4). Many values of m have been tried and we found that any values greater than or equal to 3 virtually make no difference in estimating D. Therefore, we let m equal 3 in our experiments conducted in Section 3.

The cost function, $U_i(\cdot)$, is defined as the sum of the pair-wise potentials, $W_{i,j}$, between time series i and the voting members in its voting pool, N_i

$$U_i(y_i, y_{N_i}, x_i, x_{N_i}) = \sum_{j \in N_i} W_{i,j}(y_i, y_j, x_i, x_j) \tag{9}$$

The assignment of a label $\hat{y_i}$ to a time series i is determined based on the cost function $U_i(\cdot)$ as follows

$$\hat{y_i} = \arg\min_{y_i}(U_i(y_i, y_{N_i}, x_i, x_{N_i})) \tag{10}$$

3 RESULTS AND DISCUSSION

3.1 Evaluation on Toy Cases

Since the voting pool plays a key part in determining the performance of the proposed algorithm, to demonstrate how the

Table 1. Performance of the proposed algorithm in terms of average iterations and clustering error rate with various voting pool size *k*.

k	Average iterations			Average clustering error rate (%)		
	MS & MD included	MD excluded	MS excluded	MS & MD included	MD excluded	MS excluded
4	34.50	32.56	∞	20.27	13.66	N/A
6	31.05	26.30	∞	5.10	4.24	N/A
10	16.80	21.62	483.78	0.73	0.91	0.52
18	10.75	14.60	371.33	0.47	0.40	0.33

constituent members of the voting pool affect the performance, we first tested the algorithm on various 'toy' datasets of 3-dimensional patterns consisting of 5 clusters, each having 30 patterns. The algorithm is tested with the size *k* of the voting pool set to 4, 6, 10 and 18 while the number of most-similar (*MS*) samples *s* is always fixed at 1. That is to say, for each value of *k*, there are 1 most similar (*MS*), 1 most different (*MD*), and *k* - 2 random samples in the voting pool. In order to get objective statistics during each run of the algorithm, a new sample set is used. For each sample set, the five clusters $C_i, i = 1, 2, 3, 4, 5$, are randomly created with the 5 centroids fixed at $\mu_1 = (40, 45, 100)$, $\mu_2 = (85, 60, 100)$, $\mu_3 = (65, 100, 80)$, $\mu_4 = (55, 120, 120)$, $\mu_5 = (120, 50, 120)$, respectively, and the same variance of 64. Supplementary Fig. 1 shows one of the sample set used in our experiments. The clustering performance after repeating the algorithm for 100 times is listed in Table 1. Three cases have been investigated:

1. *MS* & *MD included*: This is the case in which both the *most-similar* (*MS*) and the *most-different* (*MD*) samples are included in the voting pool.
2. *MD excluded*: This is the case in which the *MS* is included while the *MD* is not. An additional random sample is included in place of the *MD*.
3. *MS excluded*: This is the case in which the *MS* is not included while the *MD* is. An additional random sample is included in place of the *MS*.

Average iteration and *Average clustering error rate* (%) in Table 1 indicate the average iterations the algorithm have to repeat and percentage of misclassification of the 100 runs. For the first case (*MS* & *MD included*), when the number of samples *k* in the voting pool is 4, which is less than the cluster number, 5, the average iteration is as high as 34.50 and the clustering error rate is over 20%. When *k* is increased to 6, the performance improves significantly, but the error rate is still as high as 5.1%. With $k = 10$ and 18, interaction between each sample and the rest of the sample set is facilitated, as a result, the performance improves even further. One interesting point is that increasing the number of voting samples from 10 to 18 does not improve the error rate significantly. This suggests that an error rate around 0.4% to 0.8% is the best the algorithm can do, given the nature of the sample sets, and when such an error rate is achieved, including more voting samples is no longer beneficial because the overall computational load is increased, even though the average number of iterations goes down. For example, apart from other overheads, the computational cost can be measured according to the formula: $k \times$ *Average iteration*. From Table 1 we can see that the computational costs when *k* equals 10 and 18 are 168 and 193.5, respectively. Such a 0.26 % (= 0.73% - 0.47%) improvement on error rate is not a reasonable trade-off for the 15.18% (= (193.5 - 168) / 168) computational cost in most applications. The optimal size of the voting pool depends on the actual number of clusters and the total number of samples in the data set.

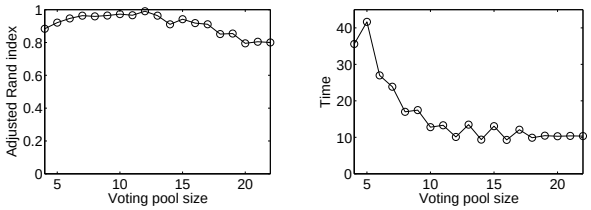
The second case (*MD excluded*) is intended to evaluate the performance of the proposed algorithm when the *most-different* sample is not involved in the voting process. When the size of the voting pool is not big enough ($k = 4$ and 6 in Table 1) to yield reasonable clustering in terms of error rate, excluding the most-different sample from the voting pool tends to have better performance in terms of *average iteration* and *error rate*. That is because the most-different sample is only updated when a more distant sample is found while the additional sample in place of the *MD* (when the *MD* is excluded) is picked at random, therefore providing more chances for interaction. However, when the size of the voting pool is big enough ($k = 10$ and 18 in Table 1) to yield reasonable clustering in terms of clustering error rate, including the most-different sample in the voting pool becomes beneficial in terms of computational cost (average iteration). The third case (*MS excluded*) is to demonstrate the importance of the *most-similar sample*. Note that the *most-different* sample can only tell the algorithm which particular label not to assign to the sample in question while the *most-similar* sample points out which label to assign. Even with the assistance of the *MD*, without the *MS*, the algorithm tends to 'guess' which label is the correct one. As indicated in Table 1, when *k* equals 4 and 6 the algorithm never converges. When *k* is increased to either 10 or 18, successful clustering become possible, but the computational cost is unacceptably higher than the cases when the *MS* is included.

We have applied *k*-means clustering to the same 5-cluster dataset of our toy case with *k* varies from 3 to 6. As expected, the *k*-means clustering algorithm always under-cluster the dataset when *k* is mistakenly set to 3 or 4, and over-cluster the dataset when *k* is mistakenly set to 6. When correct value of *k* (i.e., 5) is provided, the *k*-means clustering algorithm is still sensitive to the initial centroids. We have run *k*-means clustering algorithm with $k = 5$ for 100 times and found that the average clustering error rate is 12.9%, which suggests that the proposed CRF algorithm outperforms the *k*-means clustering algorithm when comparing to the average clustering error rate in the last two entries of Table 1.

3.2 Experiments on Time Series Datasets

From the evaluation on the above data sets, we can see that the *MS* sample plays the most important role in the clustering process. To exploit the power of similar samples in the following experiments, we set the number of *MS*, *s*, to $\lfloor \frac{k}{2} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. We use both biological datasets and simulated datasets for experimental evaluation. Biological data can only provide anecdotal evidence in clustering validation, since the knowledge of gene regulation is far from complete. Annotation information is too inconsistent to support a large scale evaluation. Simulated datasets are therefore necessary because they can provide more controllable conditions to test an algorithm and a standard for benchmarking (Ernst *et al.*, 2005; Ma *et al.*, 2006). However, to obtain meaningful results, the simulated data need to share statistical characteristics with biological data.

A popular similarity measure of two partitions, the adjusted Rand index (Hubert and Arabie, 1985), is used to assess the clustering accuracy of our algorithm. The adjusted Rand index gives the degree of agreement (in the range of 0 $\sim$ 1) between two partitions of a dataset. A high adjusted Rand index indicates a high level of agreement.



(a) (b)

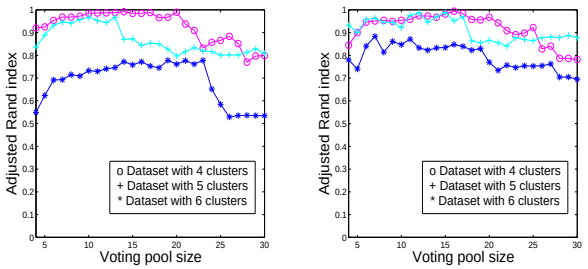
Fig. 1. Performance of our algorithm with different pool sizes for simulated dataset. Performance is evaluated in terms of (a) clustering accuracy and (b) consumed time in seconds.

3.2.1 Simulated datasets Following (Medvedovic *et al.*, 2004; Schliep *et al.*, 2005), we synthesised five classes of 60, 60, 60, 80, 60 gene expression profiles, respectively, across 20 time points according to Eq.(11), giving 320 profiles in total. Four classes are generated from sine waves, each with different frequency and phase, and the other is generated with a linear function. Gaussian noise ε_i is added to all of the data as demonstrated in Fig. 2 in the supplement.

$$\varphi(i) = \begin{cases} \sin(\alpha_i t + \beta_i) + \varepsilon_i & i \in \{1,2,3,4\} \\ a_i t + b_i + \varepsilon_i & i \in \{5\} \end{cases} \quad (11)$$

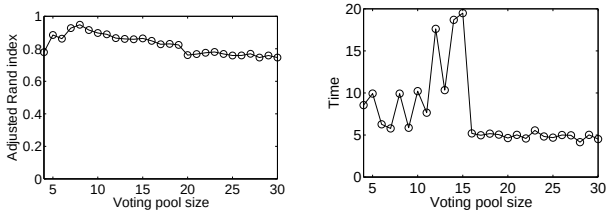
where i is the class index, α_i and β_i are frequency and phase parameters for the sine wave functions and a_i and b_i are parameters for the linear function, respectively. As discussed in (Medvedovic *et al.*, 2004; Schliep *et al.*, 2005), the mathematical model used to generate the data has been shown to have similar biological and statistical characteristics.

The simulated dataset allows us to carry out a more objective evaluation of our proposed algorithm, since the data characteristics and ground truth are known. Fig. 1(a) shows that the adjusted Rand index first climbs to a peak value of 0.9903 when the pool size, k, is 12, then it decreases and is stabilised at around 0.8 when the pool size is greater than 20. The performance of our algorithm with the voting pool size as small as 5 is already good, except that it takes the longest time to converge. A closer look at the clustering results also tells us that, with a too big voting pool, the algorithm arrives at sub-optimum results. In other words, an over-sized voting pool involves too much redundant information. As a result, distinct clusters get merged. On the other hand, the time complexity as shown in Fig. 1(b) tends to decline steadily. As noted earlier, with bigger voting pools, the iteration number drops dramatically while the time complexity increases for each iteration. Indeed, the impact on the overall time complexity due to the increased time complexity for each iteration is far less than that due to the reduced iteration count. Based on this observation, we arrive at the conclusion that the clustering accuracy and efficiency of the proposed method depend on the choice of the pool size.



(a) (b)

Fig. 2. Comparison of system performance with different voting pool sizes. Comparison of system performance with different voting pool sizes for (a) three simulated datasets with a total of 320 samples and (b) three simulated datasets with a total of 400 samples.

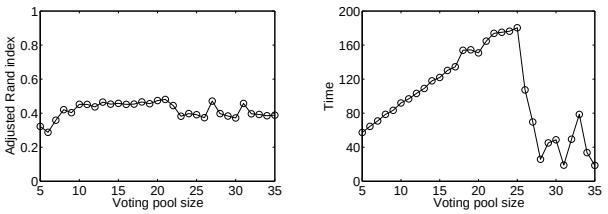


(a) (b)

Fig. 3. Performance of our algorithm with different pool sizes for Yeast Galactose dataset. Performance is evaluated in terms of (a) clustering accuracy and (b) consumed time in seconds. Each data point represents the average of ten experiments.

Further experiments were carried out for determining appropriate voting pool size. Consider six simulation set-ups: three datasets of 320 gene profiles consisting of 4, 5 and 6 clusters and four datasets of 400 gene profiles consisting of 4, 5 and 6 clusters. The data in Fig. 2 are averages of ten experimental results. In both plots we can see that the curves of the datasets with a smaller number of samples generally fall earlier than the curves of the datasets with more samples. Moreover, the performance on the dataset with 5 clusters decreases earlier than that on the dataset with 6 clusters. Since, for a dataset with only 4 clusters, the performance of our system is generally good, this may explain why the curves for the dataset with 4 clusters do not fall before the curves for the datasets with 5 clusters. However, there is no obvious pattern indicating that the actual number of clusters is an important factor in deciding the optimal voting pool size. The range of the voting pool size that gives good performance is 6 $\sim$ 12 for datasets with 320 samples and 8 $\sim$ 16 for dataset with 400 samples, respectively. We propose to select the voting pool size in proportion to the size of the dataset. Our experience suggests that a pool size in the range of 2% $\sim$ 4% of the dataset size is reasonable.

3.2.2 Yeast galactose dataset The Yeast galactose dataset (Ideker *et al.*, 2001) consists of gene expression measurements in galactose utilisation in *Saccharomyces cerevisiae*. A subset of 205 measuements of this dataset whose expression patterns reflect four functional categories as illustrated in supplementary Fig. 3 in the



(a) (b)

Fig. 4. Performance of our algorithm with different pool sizes for Y5 dataset. Performance is evaluated in terms of (a) clustering accuracy and (b) consumed time in seconds. Each data point represents the average of ten experiments.

Gene Ontology (GO) listings (Ashburner *et al.*, 2000) is used. The experiments are conducted with four repeated measurements across 20 time points. Note that the data in supplementary Fig. 3 is the mean of the four replicates. Thus, we compute the adjusted Rand index of clustering outcomes with the four functional categories as the ground truth for system validation.

We compare the adjusted Rand index obtained by our algorithm with that of the algorithm CORE proposed in a recent work (Tjaden, 2006), in which the Yeast galactose dataset is also used in experimental evaluation. Our model has a peak Rand index value of 0.9478 at the pool size of 8 as shown in Fig. 3(a), otherwise the performance is good with the pool size ranging from 5 to 11. All of them are greater than the peak value of the adjusted Rand index, roughly 0.7, reported in (Tjaden, 2006). Most importantly, our unsupervised conditional random fields model captures the correct number of clusters in the dataset, while CORE is essentially a supervised *k*-means clustering method. In terms of clustering efficiency, the clustering processes of our algorithm finishes within 30 iterations and the consumed time is less than 10 seconds in most cases, as shown in Fig. 3(b).

The performance of our method and Ng *et al.*'s mixture model-based method in terms of ARI are 0.948 and 0.978 (as reported in (Ng *et al.*, 2006)), respectively. Although Ng *et al.*'s method performs marginally better than ours, comparing to our method, the imitations of their method are: first, it assumes that the observed vectors come from a finite number of clusters/components and thus requires the user to specify the number of clusters/components. This makes the method's performance somehow dependent on the specified number of clusters. Secondly, as Ng *et al.* have mentioned that, in their model, the gene-profile vectors are not all independently distributed, they circumvent this problem by conditioning the clustering on the unobservable random gene effects and tissue effects, as they believe that given these terms, the gene-profiles are all conditionally independent (Ng *et al.*, 2006). Albeit helpful, to some extent, the performance of their method is sensitive to the specification of the random effects, which is by no means trivial. However, they did not suggest ways of specifying them.

3.2.3 Yeast cell cycle dataset In the Yeast Cell Cycle (Y5) (Cho *et al.*, 1998) dataset there are more than 6,000 yeast genes measured during two cell cycles at 17 time points. In (Schliep *et al.*, 2005), a subset of 384 genes is identified based on their peak times of five phases of the cell cycle: Early G1(G1E), late G1(G1L), S, G2 and M phase. The expression levels of each gene were standardised to enhance the performance of model-based methods. In our experiment we use the same dataset in order to compare the results. Fig. 4(a) in the supplement is the 384 time series data put together. Fig. 4(a) in the supplement (b)-(f) depict the 5 individual groups; we can see the subtle distinctions of different peak times among groups.

Table 2. Experimental results on class predictor for three datasets. Three datasets are tested in terms of the class prediction ability of the proposed algorithm. The sizes of training sets and testing sets are listed in this table. The results, Rand index and prediction error rates, are the average of results from 27 experiments.

Experiment dataset	Training samples	Testing samples	Average Rand index	Average error rate
Y5	300	84	0.4644	0.4671
Yeast Galactose	150	55	0.9318	0.0483
Simulated	260	60	0.8250	0.0653

For the Y5 dataset, we evaluated the system performance with different voting pool sizes and depict the results in Fig. 4. The best result is obtained at the pool size of 21 with a Rand index of 0.4808. As one can see from Fig. 4(a) when the pool size of our algorithm is in the range from 13 to 20, the adjusted Rand indexes are all higher than the best result, 0.467, of all of the methods without labelled data, including different HMM models (Schliep *et al.*, 2005), k-means and splines (Bar-Joseph *et al.*, 2002) evaluated in (Schliep *et al.*, 2005). Note that the proposed algorithm starts with a random label configuration, without the user providing prior knowledge. An advantage of the type of unsupervised method is that it is independent of annotation data, which are notoriously incomplete (Slonim, 2002). Meanwhile, not all unsupervised methods are able to separate genes from Late G1 and S or M and Early G1. MCLUST (Fraley and Raftery, 2002) is a widely used mixture model-based clustering method. It is unsupervised, not only in determining the number of clusters, but also in selecting the type of model that best fits the data. An R implementation of MCLUST (Fraley and Raftery, 1999) is used in our experiment. Using the EEE (Equal volume, shape and orientation) model, MCLUST yields either four or eight as the optimal number of clusters, .

To our knowledge, the clustering results on the Y5 dataset are generally not good for current methods in the literature, in contrast to those of Yeast galactose dataset by the same methods. This is because of the ambiguities among the Y5 groups. As Fig. 4(a) indicates, large pool sizes do not necessarily facilitate better performance. On the contrary, the Rand index starts to decline after a certain point. This is because of noise and outliers, which jeopardise the accuracy of the system. Moreover, bigger pool sizes give rise to higher computational cost for each iteration in our CRF model. Despite this, the decline in time complexity with increasing pool size, as shown in Fig. 4(b), is not surprising, because bigger pool sizes facilitate more global information, thus taking fewer iterations to converge.

3.3 Evaluation of class prediction

Another desirable function of a clustering algorithm, other than class discovery, is its ability to assign genes to known classes, which

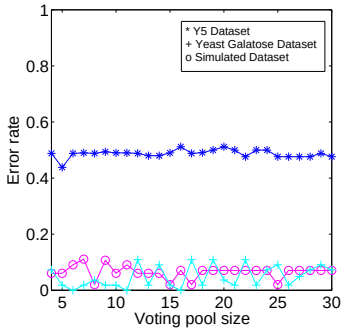


Fig. 5. Testing error rates of predictors trained by Y5 dataset, Yeast galactose dataset and simulated dataset

is also known as class prediction. Class prediction starts with the training of a class predictor, which can then be tested for its accuracy in predicting the classes of the test data. After dividing each of the above three datasets into training and testing sets as shown in Table 2, we trained our proposed clustering algorithm and then used it to predict the classes of the test data. The average prediction accuracies of 27 experiments in terms of Rand index and prediction error rates are cross-tabulated in Table 2.

The high prediction error rate for dataset Y5 is due to the high rate of misclassification of the training set, which is confirmed by a low Rand index. In contrast, the class predictions for the Yeast galactose dataset and the simulated dataset are more accurate when the predictors are well-trained as indicated in Fig. 5.

4 CONCLUSIONS

We presented an efficient unsupervised Conditional Random Fields model for both gene class discovery and class prediction, which does not require the user to provide *a priori* knowledge, such as the cluster number and the centroids. The small voting pool (compared to the size of the datasets) progressively updated in each iteration effectively facilitates both local and global interactions and overcomes the high computational complexity problem of most CRF schemes. Conditional Random Fields (CRF's) offer a desirable set of properties in gene expression time series analysis. They have great flexibility in capturing arbitrary, overlapping and agglomerative attributes of the data. Most importantly, the conditional nature of CRF's directly results in their efficiency in terms of the relaxation of independence assumptions needed by HMM's.

The biological relevance of our method was demonstrated through statistical analysis of experimental results on the Yeast galactose dataset and the Yeast cell cycle dataset. By comparing this with the results of recent work, we demonstrated the effectiveness of our CRF system.

Despite its popularity in many applications, the Rand index makes no use of the biological annotation data. This implies the development of new measure for validating gene clustering results. Although, in most cases, the accuracy and completeness of gene annotation is by no means sufficient, these annotation can serve as useful prior knowledge for validating clustering results. It is our intention to take this as a new line of investigations in the near future.

REFERENCES

Ashburner, M. *et al.* (2000). Gene ontology: tool for the unification of biology. *Nat Genet*, **25**(1), 25–29.

Bar-Joseph, Z. *et al.* (2002). A new approach to analyzing gene expression time series data. *Proceedings of the Annual International Conference on Computational Molecular Biology, RECOMB*, pages 39–48.

Boutros, P. C. and Okey, A. B. (2005). Unsupervised pattern recognition: An introduction to the whys and wherefores of clustering microarray data. *Brief Bioinform*, **6**(4), 331–343.

Cho, R. J. *et al.* (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, **2**(1), 65–73.

Culotta, A. *et al.* (2005). Gene prediction with conditional random fields.

Dojer, N. *et al.* (2006). Applying dynamic bayesian networks to perturbed gene expression data. *BMC Bioinformatics*, **7**.

Ernst, J. *et al.* (2005). Clustering short time series gene expression data. *Bioinformatics*, **21**(SUPPL. 1).

Fraley, C. and Raftery, A. (1999). Mclust: Software for model-based cluster and discriminant analysis.

Fraley, C. and Raftery, A. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, **97**(458), 611–631.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images. **6**, 721–741.

Heard, N. A. *et al.* (2006). A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: An application of bayesian hierarchical clustering of curves. *Journal of the American Statistical Association*, **101**(473), 18–29.

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

Husmeier, D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, **19**(17), 2271–2282.

Ideker, T. *et al.* (2001). Integrated Genomic and Proteomic Analyses of a Systematically Perturbed Metabolic Network. *Science*, **292**(5518), 929–934.

Ji, X. *et al.* (2003). Mining gene expression data using a novel approach based on hidden markov models. *FEBS Letters*, **542**(1-3), 125–131.

Lafferty, J. D. *et al.* (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Luan, Y. and Li, H. (2003). Clustering of time-course gene expression data using a mixed-effects model with B-splines. *Bioinformatics*, **19**(4), 474–482.

Ma, P. *et al.* (2006). A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, **34**(4), 1261–1269.

Medvedovic, M. *et al.* (2004). Bayesian mixture model based clustering of replicated microarray data. *Bioinformatics*, **20**(8), 1222–1232.

Ng, S. K. *et al.* (2006). A mixture model with random-effects components for clustering correlated gene-expression profiles. *Bioinformatics*, **22**(14), 1745–1752.

Schliep, A. *et al.* (2005). Analyzing gene expression time-courses. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **2**(3), 179–193.

Slonim, D. K. (2002). From patterns to pathways: gene expression data analysis comes of age. *Nature Genetic*, **32 Suppl**, 502–508.

Tjaden, B. (2006). An approach for clustering gene expression data with error information. *BMC Bioinformatics*, **7**, 17.

Wu, F. X. *et al.* (2005). Dynamic model-based clustering for time-course gene expression data. *Journal of Bioinformatics and Computational Biology*, **3**(4), 821–836.