

Clustering Gene Expression Series with Prior Knowledge

Laurent Bréhélin

Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier
161, rue Ada, 34392 Montpellier Cédex 5, France
brehelin@lirmm.fr

Abstract. Microarrays allow monitoring of thousands of genes over time periods. Recently, gene clustering approaches specially adapted to deal with the time dependences of these data have been proposed. According to these methods, we investigate here how to use prior knowledge about the approximate profile of some classes to improve the classification result. We propose a Bayesian approach to this problem. A mixture model is used to describe and classify the data. The parameters of this model are constrained by a prior distribution defined with a new type of model that can express both our prior knowledge about the profile of classes of interest and the temporal nature of the data. Then, an EM algorithm estimates the parameters of the mixture model by maximizing its posterior probability.

Supplementary Material:

<http://www.lirmm.fr/~brehelin/WABI05.pdf>

1 Introduction

Technological advances such as microarrays allow us to simultaneously measure the level of expression of thousands of genes in a given tissue over time—for example along the cell cycle [1]. In the following, such a series of gene expression measurements is called an *expression series*. One common problem of gene expression data analysis is the identification of co-regulated genes. This problem naturally turns into a gene clustering problem. Until recently, expression series have been analyzed with methods that do not take the time dependences into account. Such methods include hierarchical clustering with Euclidean distance [2], k-means approaches [3, 4] and the Self Organizing Maps [5, 6]. Since these methods are unable to explicitly deal with the data order, permuting two or more time points in all series does not change the clustering result. A few methods specially adapted to expression series have recently been proposed. These methods involve probabilistic modeling of the data. For example, [7] use autoregressive models of order p . [8] use cubic splines with a probabilistic component to model the classes, while [9] model each class of gene with Hidden Markov Models (HMMs) [10].

Our aim here is to investigate how to explicitly use *rough* prior knowledge about the general shape of interesting classes. By *general shape*, we mean elementary and potentially incomplete information about the evolution of the mean

expression level of the classes over time. This can, for example, be knowledge like: “Classes with increasing expression level”, “Classes with bell curve shapes”, “Classes with high expression level in the beginning of the series”, etc. Of course we do not know the profile of *all* the gene classes, but sometimes we are more concerned with one or more classes. For example, in the study of [1] on the Yeast cell cycle, the authors are interested in finding the cycle-regulated genes, and thus look for sinusoidal shape classes. In a similar way, we sometimes search for genes which tend to be quickly over- (or under-) expressed at the beginning of the series—in response to a given treatment, for example. A problem of importance that arises when the awaited classes are sparse—i.e., there are few interesting genes with regards to all the other ones—is that standard methods can completely omit these classes. This results in a final clustering where the interesting genes are lost among many other genes, in one or more classes that do not show the desired profile.

The approach we propose here tackles this problem. When information about one or several class shapes are available, these are directly integrated into the model, thus favoring classes with the desired profiles, and putting the other genes in separate classes. On the other hand, when no a priori information is available, the method allows a classical clustering of the series that deals with the temporal nature of the data in a very intuitive way. We use a Bayesian approach for this purpose. The approach involves two types of models. The first one is a probabilistic mixture model used to describe and classify the expression series. Parameters of this model are unknown and have to be estimated for the clustering. A second model, close to the HMMs and called *HPM*—for *Hidden Phase Model*—, is used to express our a priori knowledge (or simply the temporal feature of the data). We define two types of HPMs which can be used according to the situation: probabilistic and non-probabilistic HPMs. These models are completely specified by the user, and their parameters do not have to be estimated. They are used to define a prior probability distribution over the parameters of the mixture model. These parameters are estimated by maximizing the posterior probability of the model through an EM algorithm [11].

The next section presents our method, the mixture model, the two types of HPMs and the learning algorithm. In Section 3 we evaluate our method. We conclude in Section 4.

2 Method

2.1 Principle

Let \mathcal{X} be a set of N expression series of length T . We assume that the data arise from a mixture model [12] with C components. We denote π_c as the prior probability of component c , and we have $\sum_{c=1}^C \pi_c = 1$. We assume that conditionally to component c , expression values at each time $t \in [1, T]$ are independent and follow a Gaussian distribution of mean μ_{ct} and variance σ_{ct}^2 . The shape of component c is defined by the sequence of means $\mu_{c1} \dots \mu_{cT}$.

We then have a probabilistic model of parameters $\Theta = (\pi_1, \dots, \pi_C, \theta_1, \dots, \theta_C)$ with $\theta_c = (\mu_{c1}, \dots, \mu_{cT}, \sigma_{c1}^2, \dots, \sigma_{cT}^2)$. The probability of an expression series $X = x_1 \dots x_T$ in this model is

$$P(X|\Theta) = \sum_{c=1}^C \pi_c \prod_{t=1}^T P(x_t|\mu_{ct}, \sigma_{ct}^2),$$

with $P(x_t|\mu_{ct}, \sigma_{ct}^2) = \mathcal{N}(x_t; \mu_{ct}, \sigma_{ct}^2)$. Under the assumption that series of \mathcal{X} are independent, the likelihood of Θ is given by

$$L(\Theta|\mathcal{X}) = P(\mathcal{X}|\Theta) = \prod_{X \in \mathcal{X}} P(X|\Theta). \quad (1)$$

In a clustering task, the standard approach to classify a set of expression series \mathcal{X} involves estimating parameters Θ that maximize Formula (1) (Maximum Likelihood Principle), and then assigning the most probable component c_{MAP} (*MAP* stands for *maximum a posteriori*) to each series $X \in \mathcal{X}$:

$$c_{\text{MAP}} = \operatorname{argmax}_{c=1 \dots C} P(c|X, \Theta) = \operatorname{argmax}_{c=1 \dots C} \pi_c P(X|c, \Theta) \quad (2)$$

Note that finding parameters Θ that maximize (1) is a difficult task. However, approximate solutions can be inferred with EM algorithms [11].

The above mixture model does not explicitly take into account the potential dependences between times, nor any prior knowledge about the profile of the most interesting classes. Our aim is to constraint one or some components to follow a given profile, while leaving the other components free of constraints so that they can “collect” the expression series that do not have the desired profile. For example, if we are looking for classes with bell curves, we would build a 10 component model, with 5 bell-constrained and 5 unconstrained components. We thus propose to use a Bayesian approach, which introduces knowledge by way of a prior distribution of Θ —see for example [13] for a general introduction to Bayesian theory. Simply speaking, our idea is to define a prior distribution $P(\Theta)$ which is merely the product of the prior probability of the sequences of means $\mu_{c1} \dots \mu_{cT}$ associated with each component. Moreover, we want the prior probability of a given mean sequence for component c as follows: (i) the more the sequence agrees with the constraints associated with c , the higher its prior probability; (ii) sequences that disagree with the constraints have probability zero.

With a prior, we can write the posterior probability of Θ as

$$P(\Theta|\mathcal{X}) = \frac{P(\mathcal{X}|\Theta)P(\Theta)}{P(\mathcal{X})} \propto P(\mathcal{X}|\Theta)P(\Theta). \quad (3)$$

In this Bayesian framework, parameters Θ are estimated by maximizing the posterior probability —Equation (3)— instead of the likelihood —Expression (1). However, maximizing the posterior probability is generally more difficult than maximizing the likelihood. For example, the classical re-estimation formulae of

the EM algorithm do not directly apply and, depending on the form of the chosen prior distribution, it may be hard to perform the task in reasonable time.

In our case, we first discretize the space of the means μ_{ct} in order to be able to introduce various bits of knowledge and constraints about the profiles, as well as to efficiently estimate the parameters of the model. Since we know the maximal and minimal expression values taken by the series in \mathcal{X} (say x_{\max} and x_{\min}), we already know an upper and lower bound of the space of the means. Now we discretize this space in M equidistant steps, so that the lower and higher steps are equal to x_{\min} and x_{\max} , respectively. Of course M is chosen to be sufficiently large (e.g. $M = 30$) to allow accurate representation of the data. Steps are named by their number, so M is the highest step. In this discretized mean space, our probabilistic model is re-defined as $\Theta = (\pi_1, \dots, \pi_C, \theta_1, \dots, \theta_C)$ with $\theta_c = (l_{c1}, \dots, l_{cT}, \sigma_{c1}^2, \dots, \sigma_{cT}^2)$, with $l_{ct} \in \{1, \dots, M\}$. We denote $m : \{1, \dots, M\} \rightarrow [x_{\min}, x_{\max}]$ as the map function that associates step l with its expression level. The probability of an expression series $X \in \mathcal{X}$ is rewritten as

$$P(X|\Theta) = \sum_{c=1}^C \pi_c \prod_{t=1}^T P(x_t|l_{ct}, \sigma_{ct}^2),$$

with $P(x_t|l_{ct}, \sigma_{ct}^2) = \mathcal{N}(x_t; m(l_{ct}), \sigma_{ct}^2)$ that follows a Gaussian distribution of mean equal to the level of expression associated with step l_{ct} , and variance σ_{ct}^2 . In the following, the step sequence $l_{c1} \dots l_{cT}$ associated with class c —and which defines its shape—is denoted as L_c . Note finally that the discretization only involves the means of the model, and not the space of the expression levels of the data. These, as well as the model variances σ_{ct}^2 , remain in a continuous space.

2.2 Defining the prior distribution

First we define a new type of model called *Hidden Phase Models* (or *HPMs*), close to models like HMMs and finite automata [14]. These HPMs are used to express the desired profiles of the components, and each component c is then associated with a given HPM H_c . We define two types of HPMs: probabilistic and non-probabilistic HPMs. We next show how to derive the prior distribution of Θ from the HPMs.

Hidden Phase Models The general assumption behind HPMs is that the genes of a given component pass through *phases* or *biological states* over the time. This means that, for a given component, we assume that some ranges of consecutive times actually correspond to the same biological state. These phases are hidden, but they affect the mean expression level evolution of the component. For example, some phases induce an increase in the mean level expression level while others tend to decrease or stabilize the level. In the same manner, the increase (or decrease) can be high for some phases and low for others, etc.

A (non-probabilistic) HPM is defined by a quadruplet $(\mathcal{S}, \delta, \epsilon, \tau)$, where

- \mathcal{S} is a set of states representing the different phases; \mathcal{S} contains two special states, *start* and *end*, which are used to initiate and conclude a sequence, respectively.
- $\delta : \mathcal{S} \times \mathcal{S} \mapsto \{0, 1\}$ is a function describing the authorized transitions between states. We denote $Out(s)$ as the set of states that can be reached from s .
- ϵ is a function that associates each state $s \in \mathcal{S}$ with an interval of integers defining the minimal and maximal differences of steps that can be observed between times t and $t - 1$ when genes are in state s at time t . For example, if $\epsilon(s) = [1, 3]$, this means that if the genes of the component are in phase s at time t then the step difference ($l_t - l_{t-1}$) is between 1 and 3 (so phase s increases the expression level).
- τ is a function that associates each state $s \in \mathcal{S}$ with the interval of time the state can be reached. For example, if $\tau(s) = [3, 5]$ then the genes can be in state s between times 3 and 5 included.

An HPM example is depicted in Figure 1.

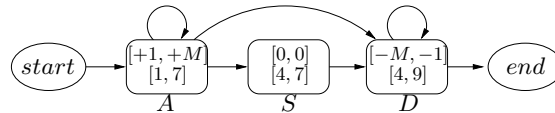


Fig. 1: An HPM for clustering 9-time expression series. In each state, upper and lower intervals represent the step-difference and time intervals associated with the state, respectively. This HPM induces bell curve shapes.

Now we can see how to express our prior knowledge with an HPM. Actually an HPM defines a set of *compatible* step sequences. We say that a step sequence $L = l_1 \dots l_T$ is compatible with an HPM H if there is a state sequence $s_0 \dots s_{T+1}$ —with $s_0 = start$ and $s_{T+1} = end$ —in H , which is compatible with L . And we say that a state sequence $s_0 \dots s_{T+1}$ is compatible with L iff for each time $1 \leq t \leq T$ we have: i) t included in the time interval $\tau(s_t)$; ii) $\forall t \geq 2$, $(l_t - l_{t-1})$ included in $\epsilon(s_t)$ —for $t = 1$, as we do not know l_0 , the genes can be in any phase so s_1 can be any state. Considering the step sequence on the top of Figure 2, a compatible phase sequence in the HPM of Figure 1 is, for example, *start* – *A* – *A* – *A* – *A* – *S* – *D* – *D* – *D* – *D* – *end*. For the step sequence on the right, there is no compatible phase sequence in this HPM. In brief, building an HPM involves designing an HPM such that the compatible sequences have the desired profile. For example, the HPM of Figure 1 is well suited for the discovery of bell curve classes.

Probabilistic HPMs Non probabilistic HPMs can be used to express strong constraints. They are generally sufficient to express knowledge about simple or well defined profiles. For more complex knowledge, or when we do not have any information about profiles and just want to express the fact that we are dealing with series data, these models can be unsuitable. Then probabilistic HPMs can be more suitable.

A probabilistic HPM is defined by a quintuplet $(\mathcal{S}, \delta, \epsilon, \tau, w)$, where \mathcal{S} , δ , ϵ , and τ are the same as for non-probabilistic HPMs, and $w : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}^+$ is a function associating a weight with each authorized transition. These weights are used to compute the transition probabilities from state to state. Due to the time constraints associated with the states by way of the τ function, transition probabilities are time dependent, so we cannot simply label transitions with a probability as is done for classical HMMs. In contrast, the probability, denoted as $P(s|s', t)$, to reach state s from state s' at time t is computed as follows:

$$P(s|s', t) = \begin{cases} 0 & \text{if } t \notin \tau(s); \\ w(s) / \left(\sum_{s'' \in \text{Out}(s') \mid t \in \tau(s'')} w(s'') \right) & \text{else.} \end{cases} \quad (4)$$

One example of probabilistic HPM is depicted in Figure 2.

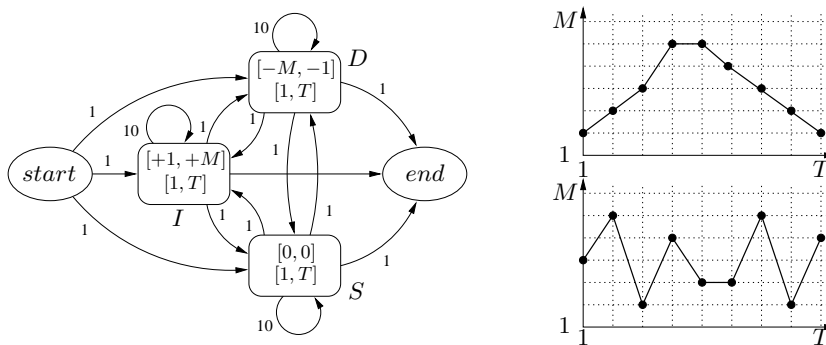


Fig. 2: Left, a probabilistic HPM for clustering expression series without prior knowledge about the form of the profiles. Right, two examples of step sequences.

Probabilistic HPMs also define compatible step sequences. Moreover, all compatible sequences do not have the same probability. Let H be a probabilistic HPM and $S = s_0, s_1 \dots s_T, s_{T+1}$ a state sequence in this HPM. The probability of this sequence given H is defined by

$$P(S|H) = \prod_{t=1}^{T+1} P(s_t | s_{t-1}, t). \quad (5)$$

This model enables us to introduce more knowledge about the desired classes. For example, when we do not have any information about interesting profiles, the only thing we know is that we have to classify expression series. This means that we are seeking relatively “regular” profiles, in contrast to chaotic spiky profiles as that depicted on the bottom of Figure 2. This knowledge can be easily expressed with the probabilistic three-states HPM of Figure 2: one state (I) induces increasing steps, one (D) induces a decrease, and the last (S) induces stability. Moreover, it is assumed that, at each time, the probability of staying in the same state is higher than the probability of departure from it (weights on loops are higher than on other transitions). This HPM is compatible with

any step sequence of length 9. However all sequences do not have the same probability, and spiky sequences involving many state changes are not favored.

Note that given a step sequence L , there are potentially many state sequences compatible with L . In reference to the HMM literature, the sequence of phases compatible with L which has the highest probability is called the *Viterbi sequence* of L [10], and is denoted as $V^L = v_0^L \dots v_{T+1}^L$. For example, the Viterbi sequences of the two step sequences of Figure 2 in the HPM of Figure 2, are *start - I - I - I - I - S - D - D - D - D - end* and *start - I - I - D - I - D - S - I - D - I - end*, respectively.

Defining prior with HPMs First we assume that prior probabilities of parameters π_c , L_c and σ_{ct}^2 are independent, as well as the C sets of parameters L_c and $(\sigma_{c1}^2, \dots, \sigma_{cT}^2)$, i.e., the probability distribution can be written as:

$$P(\Theta) = P(\pi_1, \dots, \pi_C) \prod_{c=1}^C P(L_c) \prod_{c=1}^C P(\sigma_{c1}^2, \dots, \sigma_{cT}^2).$$

Next we assume that distributions $P(\pi_1, \dots, \pi_C)$ and $P(\sigma_{c1}^2, \dots, \sigma_{cT}^2)$ are uninformative and that probabilities $P(L_c)$ are the only ones that express our knowledge.

Let c be a component and H_c a non probabilistic HPM associated with this class. A prior distribution of parameters L_c can be defined with H_c by assuming that the step sequences incompatible with H_c have probability zero while compatible sequences have all the same probability, i.e.,

$$P(L|H_c) = \begin{cases} 0 & \text{if } L \text{ is incompatible with } H_c; \\ K_c & \text{else,} \end{cases} \quad (6)$$

with K_c such that $\sum_{L \in \mathcal{L}_T} P(L) = 1$, with \mathcal{L}_T being the set of length T sequences.

For probabilistic HPM, we want the prior probability of a step sequence L to be proportional to the Viterbi sequence of L in H_c . Then, we set

$$P(L|H_c) = \begin{cases} 0 & \text{if } L \text{ is incompatible with } H_c; \\ K'_c \cdot P(V^L|H_c) & \text{else,} \end{cases} \quad (7)$$

with K'_c such that $\sum_{L \in \mathcal{L}_T} P(L) = 1$. For example, for the HPM of Figure 2, the prior probabilities of the two step sequences are proportional to $1/3 \cdot .7 \cdot .7 \cdot .7 \cdot .1 \cdot .1 \cdot .7 \cdot .7 \cdot .7 \cdot .1 \sim 3.9 \cdot 10^{-5}$ and $1/3 \cdot .7 \cdot .1 \cdot .1 \cdot .1 \cdot .1 \cdot .1 \cdot .1 \cdot .1 \cdot .1 \sim 2.3 \cdot 10^{-10}$, respectively. The spiky sequence is then less likely than the other one, which agrees with our prior intuition.

A prior distribution of the step sequences of length T can then be defined with a probabilistic or a non-probabilistic HPM. In practice, one or more components can be associated with a given HPM (e.g. that of Figure 1), and the other ones with a less informative HPM like that of Figure 2. We then have

$$P(\Theta) \propto \prod_{c=1}^C P(L_c|H_c). \quad (8)$$

2.3 Learning

Here we briefly describe the learning algorithm used to estimate parameters Θ of the mixture model. A more detailed version can be found in the supplementary information material¹. It is an EM algorithm that searches for parameters that maximize Expression (3). We only give the algorithm used for probabilistic HPMS, since that for non-probabilistic ones can be easily adapted.

Let us first define the *complete-data* likelihood. Likelihood of Expression (1) is actually the incomplete-data likelihood, since the real components of series $X \in \mathcal{X}$ are unknown. Under the assumption that this set of components $\mathcal{C} = \{c_X \in \{1, \dots, C\}, \forall X \in \mathcal{X}\}$ is known, the complete-data likelihood can be written as

$$L(\Theta|\mathcal{X}, \mathcal{C}) = P(\mathcal{X}, \mathcal{C}|\Theta) = \prod_{X \in \mathcal{X}} \pi_{c_X} \prod_{t=1}^T P(x_t; l_{c_X t}, \sigma_{c_X t}^2).$$

The EM algorithm is an iterative algorithm that starts from an initial set of parameters $\Theta^{(0)}$, and iteratively reestimates the parameters at each step of the process. Let $Q(\Theta, \Theta^{(i)})$ denote the expectation, on the space of the hidden variables \mathcal{C} , of the logarithm of the complete-data likelihood, given the observed data \mathcal{X} and parameters $\Theta^{(i)}$ at step i :

$$Q(\Theta, \Theta^{(i)}) = \sum_{\mathcal{C} \in \mathbf{C}} \log P(\mathcal{X}, \mathcal{C}|\Theta) P(\mathcal{C}|\mathcal{X}, \Theta^{(i)}),$$

with \mathbf{C} being the space of values \mathcal{C} can take. From [11], one can maximize Expression (3) by searching at each step of the algorithm for parameters π_c^* , L_c^* and σ_{ct}^{2*} that maximize the quantity

$$Q(\Theta, \Theta^{(i)}) + \log P(\Theta). \quad (9)$$

Since $P(\Theta)$ is not related to the parameters π_c , after some calculus, an expression can be derived for π_c^* that maximizes Expression (9):

$$\pi_c^* = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} P(c|X, \Theta^{(i)}). \quad (10)$$

Now, due to our independence assumptions, one can estimate the L_c and σ_{ct}^2 that maximize Expression (9) for each component c independently. As for parameters π_c , σ_{ct}^2 are not involved in the expression of $P(\Theta)$. Moreover, since the σ_{ct}^2 associated with time t is independent of all the other times, the expression of σ_{ct}^{2*} that maximizes (9) depends solely on the step l_{ct}^* in L_c^* :

$$\sigma_{ct}^{2*} = \frac{\sum_{X \in \mathcal{X}} (x_t - m(l_{ct}^*))^2 P(c|X, \Theta^{(i)})}{\sum_{X \in \mathcal{X}} P(c|X, \Theta^{(i)})}. \quad (11)$$

¹ <http://www.lirmm.fr/~brehelin/WABI05.pdf>

For L_c the situation is quite different since it is involved in the expression of $P(\Theta)$. The L_c that maximizes Expression (9) depends both on the data and on its Viterbi path in H_c and hence the different steps l_{ct}^* of L_c^* cannot be estimated independently. However, the step space is of finite size, so the space of the step sequences of length T is also finite. One way to compute the new L_c would be to enumerate all possible step sequences and then select the one that maximizes Expression (9). However, as the total number of length T sequences is equal to M^T , enumerating them all is clearly not suitable. Instead, we use a dynamic programming approach that iteratively computes the best sequence without enumerating all the solutions. Briefly, for each step l and each time t , we compute iteratively, from $t = 1$ to T , the best sequence —with regard to Expression(9)— that ends on step l at time t . At each iteration and for each step l , this best sequence is computed using the results of the previous iteration, and at the end of the process the best sequence L_c^* has thus been computed in polynomial time.

The learning algorithm is depicted in Figure 3. When no better solution is available, the initial parameter values can be set randomly. Thanks to the EM properties, the posterior probability $P(\Theta|\mathcal{X})$ —and hence $P(\mathcal{X}|\Theta)P(\Theta)$ — increases at each loop of the algorithm, until a local optimum is reach. Then it continues to increase but to a much lesser extent. A practical way to detect the convergence is to check the increase at each loop and to stop the algorithm when this value goes under a given boundary.

```

1 Set parameters to initial values
2 repeat
3   for  $c = 1$  to  $C$  do
4     compute  $\pi_c^*$  with Formula (10)
5     Find the optimal step sequence  $L_c^* = l_{c1}^* \dots l_{cT}^*$  with the dynamic
       programming algorithm
6     foreach time  $t$  do compute  $\sigma_{ct}^{2*}$  from  $l_{ct}^*$  with Formula (11)
7   Compute  $P(\mathcal{X}|\Theta)P(\Theta)$ 
8 until convergence

```

Fig. 3: Learning algorithm

The total time complexity of the learning algorithm is $O(BCTM^2R^2N)$ —see supplementary information for details—, with B , C , T , M , R and N the maximal number of loops of the EM algorithm, the number of components of the mixture model, the number of time points of the data, the size of the step space, the maximal number of states of the HPMS, and the number of expression series to classify, respectively. In practice, N is potentially high (some thousands), T and R are relatively low (ten or less), M is around thirty, and less than one hundred loops are generally sufficient to ensure convergence. For the experiments in the next section for example, computing times on a 2 GHz Pentium 4, range from 20 seconds to 3 minutes according to the dataset, the type of HPMS and the number of components.

3 Experiments

In order to quantify the advantages of using prior knowledge to recover a particular class of genes, we first conducted some experiments on a dataset made up of the original Fibroblast dataset of [15] (see Supplementary Information for more details), along with some additional synthetic series that form a new artificial class. Briefly, we use a probabilistic model involving two Gaussian distributions to generate the expression levels of the artificial expression series: one Gaussian distribution is used to independently generate the gene expression levels of the first three times, while the other is used for the last nine times of the series. The mean of the first one is higher than the second, so the shape of the artificial class looks like a descending step. Figure 4 shows an example of synthetic series generated with this model. We conducted several experiments to recover the synthetic class among all other series, with the proportion of synthetic data ranging from 2% to 16% of the total data.

We use two quantities to measure the ability to recover the artificial class in the final clustering: *Recall* is the highest proportion of this class that can be found in a single cluster —so a recall of 100% is achieved when all the artificial series are in the same cluster—, and *precision* represents the proportion of artificial series in this cluster —so a precision of 100% indicates that all the series in the cluster containing most artificial series are actually artificial. For each proportion of synthetic data, we run a clustering of 11 components with two different methods. The first one does not use any prior knowledge about the class of interest, i.e., its components are completely unconstrained —this method can be viewed as a kind of k-means clustering. The second method makes use of the HPM of Figure 4 to constrain the first class, leaving the 10 others unconstrained. The experiments were repeated 100 times for each proportion of synthetic data and the results are reported in Figure 5.

Both methods achieve quite good recall, even when the proportion of the class of interest is low. Using prior knowledge gives only slightly better results. Concerning the precision, however, there is a clear difference between the two methods, and we can see that the lower the proportion of interesting class, the higher the benefit of our method. When the proportion is 2%, for example, the precision achieved with no prior knowledge is only about 21% —vs. 65% when using prior knowledge—, so the interesting series are lost among many other series, leading to a class that does not show the desired profile.

Next we investigated the sensitivity of the method to the number of components. Determining the number of clusters is a difficult task for all clustering methods. However, when the aim is to recover a particular class of genes rather than to infer a global clustering of the data, the problem is less acute. To illustrate this, we computed, in 100 runs, the precision and recall achieved with various numbers of constrained and unconstrained components, with the proportion of synthetic data ranging from 2% to 16% of the total data. We tried 1 constrained with 8, 10, 12 and 15 unconstrained components, and 2 constrained with 10 unconstrained components. All trials gave recall of up to 80% for all proportions of synthetic data (data not shown), and quite good precision —

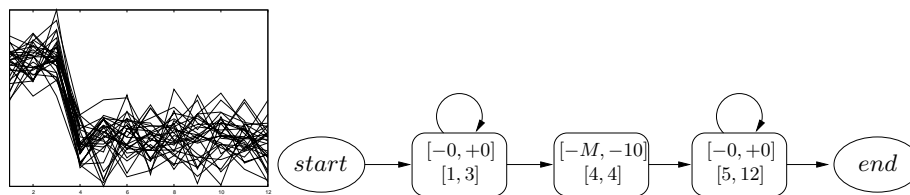


Fig. 4: Left, examples of synthetic expression series added to the fibroblast dataset. Right, the HPM designed to find the synthetic class among the "real" biological classes in the fibroblast dataset.

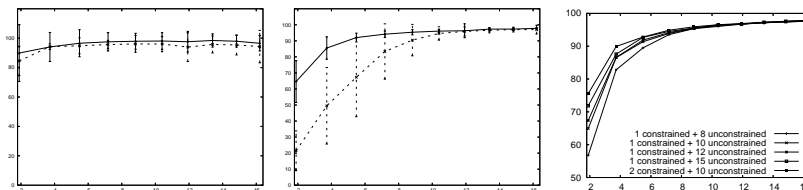


Fig. 5: Recall (left) and precision (middle) achieved with (solid lines) and without (dashed lines) prior knowledge about the class of interest. The x-axes denote the proportion (in percent) of this class among all the expression series. Right, precision achieved using different number of components.

see the right hand curves in Figure 4. Actually the best results are achieved with the highest numbers of components, so giving a sufficiently high number of components seems to be a good strategy to efficiently recover the clusters of interest.

Experiments to find "real" classes have also been carried out. We used the datasets of [15] and [1] with the aim to uncover classes that show a quick over-expression at the beginning of the series and classes with sinusoidal shape, respectively. Due to space limitations, these experiments have been included in Supplementary Material.

4 Conclusions

We proposed a Bayesian approach for the clustering of gene expression series. This approach allows the user to easily integrate prior knowledge about the general profile of the classes of interest.

We experimentally observed on a mixture of natural and synthetic data that the benefit of the method increases when the number of expression series composing the classes of interest decreases with regard to the total number of series, and that it can be really interesting when this number is very low.

Many improvements seem possible on this basis. Indeed, other knowledge can be integrated in the HPMs. For example, knowledge about the desired mean

expression level—and not about the *evolution* of the expression has it is done—could be easily added. Another improvement would be to introduce long-range dependences, i.e., to constrain differences of expression not only between consecutive times but also between separate times. For example, this would allow us to stipulate that the profiles should achieve their maximum at a specific time t .

Acknowledgements

I thank Olivier Martin, Gilles Caraux and Olivier Gascuel for their help and comments on this work.

References

1. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell* **9** (1998) 3273–3297
2. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A* **95** (1998) 14863–14868
3. Lloyd, S.: Least squares quantization in PCM. *IEEE Trans. Info. Theory* **IT-2** (1982) 129–137
4. Herwig, R., Poustka, A.J., Muller, C., Bull, C., Lehrach, H., O'Brien, J.: Large-scale clustering of cDNA-fingerprinting data. *Genome Res* **9** (1999) 1093–105
5. Kohonen, T.: *Self-Organizing Maps*. Springer (1997)
6. Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S., Golub, T.R.: Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc Natl Acad Sci U S A* **96** (1999) 2907–2912
7. Ramoni, M.F., Sebastiani, P., Kohane, I.S.: Cluster analysis of gene expression dynamics. *Proc Natl Acad Sci USA* **99** (2002) 9121–9126
8. Bar-Joseph, Z., Gerber, G.K., Gifford, D.K., Jaakkola, T.S., Simon, I.: Continuous representations of time-series gene expression data. *J Comput Biol* **10** (2003) 341–356
9. Schliep, A., Schonhuth, A., Steinhoff, C.: Using hidden markov models to analyze gene expression time course data. *Bioinformatics* **19 Suppl 1** (2003) 255–263
10. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77** (1989) 257–285
11. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B* **39** (1977) 1–38
12. McLachlan, G., Krishnan, T.: *Finite mixture models*. John Wiley (2000)
13. Duda, R., Hart, P., Stork, D.: *Pattern Classification*. John Wiley (2001)
14. Casacuberta, F.: Some relations among stochastic finite state networks used in automatic speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12** (1990) 691–695
15. Iyer, V.R., Eisen, M.B., Ross, D.T., Schuler, G., Moore, T., Lee, J.C., Trent, J.M., Staudt, L.M., Hudson, J.J., Boguski, M.S., Lashkari, D., Shalon, D., Botstein, D., Brown, P.O.: The transcriptional program in the response of human fibroblasts to serum. *Science* **283** (1999) 83–87