A Bayesian Approach for the Clustering of Short Time Series

Laurent Bréhélin

Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier
UMR CNRS 5506
161, rue Ada, F-34392 Montpellier Cedex 5
brehelin@lirmm.fr

ABSTRACT. *Microarrays allow monitoring of thousands of genes over time periods. However, due to the low number of time points of the gene expression series, taking the temporal dependences into account when clustering the data is an hard task. Moreover, classes very interesting for the biologist, but sparse with regard to all the other genes, can be completely omitted by the standard approaches. We propose a Bayesian approach for 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 expresses our prior knowledge. These knowledge allow to take the temporal dependences into account in natural way, as well as to express rough temporal profiles about classes of interest.*

RÉSUMÉ. *Des technologies récentes telles que les* microarrays *permettent de mesurer le niveau d'expression de milliers de gènes au cours du temps. Cependant, le nombre de points réduit de ces séries temporelles rend difficile la prise en compte des dépendances entre les temps par les algorithmes de classification. De plus, certaines des classes les plus intéressantes pour le biologiste peuvent être totalement omises par les algorithmes classiques du fait du faible nombre de gènes qui les composent. Nous proposons une approche bayésienne de ce problème. Un modèle de mélange est utilisé pour décrire et classer les données. Les paramètres de ce modèle sont contraints par une distribution* a priori *définie grâce à un nouveau type de modèles qui exprime les connaissances* a priori *dont on dispose. Ces connaissances permettent de traiter les dépendances temporelles d'une manière très naturelle, et de prendre en compte des connaissances approximatives concernant les profils temporels les plus intéressants.*

KEYWORDS: *time series, clustering, EM algorithm, bioinformatics, gene expression data.*

MOTS-CLÉS : *séries temporelles, classification non supervisée, algorithme EM, bioinformatique, données d'expression de gènes.*

RSI - RIA - 20/2006. New Methods in Machine Learning, pages 697 to 716

1. Introduction

Technological advances such as microarrays allow us to simultaneously measure the level of expression of thousands of genes in a given tissue at a given moment. These measurements can be repeated on different tissues, different biological organisms, or at different times during the life of the same organism to constitute a collection of gene expression measurements. These collections are a unique material for understanding various cellular regulation mechanisms. These collections are either ordered or non-ordered. A non-ordered collection may, for example, be a set of measurements on different patients with a given form of cancer (Alon *et al.*, 1999), or on plants growing on different substrates. Ordered collections generally consist of series of gene expressions measured over a time course —for example along the cell cycle (Spellman *et al.*, 1998). The order is generally defined by time, but it may also be induced by other numerical features. In (Hertzberg *et al.*, 2001) for example, the expression levels are measured at different depths of the stem of poplar trees. In other studies, measurements are obtained on cells exposed to increasing concentrations of a given factor (light, chemical product, etc). In the following, such a series of gene expression measurements is called an *expression series*, and we speak about the different *time points* of the series, even if the order is not temporal.

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 (Eisen *et al.*, 1998), k-means approaches (Lloyd, 1982) and the Self Organizing Maps (Kohonen, 1997; Tamayo *et al.*, 1999). 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 generally involve probabilistic modeling of the data. For example, (Ramoni *et al.*, 2002) use autoregressive models of order p. Briefly, they assume that the value of an expression series at time t is a linear function —with a probabilistic component— of the values taken at the p previous times. (Bar-Joseph *et al.*, 2003) use cubic splines with a probabilistic component to model the classes, while (Schliep *et al.*, 2003) model each class of gene with Hidden Markov Models (HMMs) (Rabiner, 1989). However, the great majority of the expression series data have a low number of time points (10 or less), and, as stressed in (Ernst *et al.*, 2005), all these methods involve high number of parameters and then are not appropriate for short time series. In (Ernst *et al.*, 2005) authors propose a non parametric approach to the problem. The method involves enumerating a set of authorized profiles, and then associates each series with its most likely. The set of profiles is defined by a parameter c which represents the amount of change a gene can exhibit between successive time points. Thus, the time dependence is controlled by way of this parameter.

Our aim in this paper is twofold. First we are looking for a probabilistic method able to take the time dependences of these short time series into account. Second we 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 (Spellman *et al.*, 1998) 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. Our idea is that incorporating such (even rough) knowledge can improve the clustering result, especially when the classes of interest are very sparse with regard to all the other genes. A problem of importance that arises when the expected 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 these problems. 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. This is done by explicitly dealing with the temporal nature of the data, in a very intuitive way and without any assumption about a predetermined analytical form which can be difficult to estimate with low number of time points.

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 (Dempster *et al.*, 1977).

The next section presents our method, the mixture model, the two types of HPMs and the learning algorithm. In Section 3 we evaluate and experiment our method on two datasets. We conclude and propose future work directions 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 (McLachlan *et al.*, 2000) 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 $X \in \mathcal{X}$:

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

Note that finding parameters Θ that maximize (1) is a difficult task. However, approximate solutions can be inferred with EM algorithms (Dempster *et al.*, 1977).

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. When such knowledge are available, we would like 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 (Duda *et al.*, 2001) 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 $\Theta = (\pi_1, \ldots, \pi_C, \theta_1, \ldots, \theta_C)$ with $\theta_c = (l_{c1}, \ldots, l_{cT}, \sigma^2_{c1}, \ldots, \sigma^2_{cT})$, with $l_{ct} \in \{1, \ldots, M\}$. We denote m as the map function that associates step $l \in \{1, \ldots, M\}$ with its expression level in the interval $[x_{\min}, x_{\max}]$. The probability of an expression series $X \in \mathcal{X}$ is rewritten as

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

with $P(x_t|l_{ct}, \sigma^2_{ct}) = \mathcal{N}(x_t; m(l_{ct}), \sigma^2_{ct})$ that follows a Gaussian distribution of mean equal to the level of expression associated with step l_{ct}, and variance σ^2_{ct}. In the following, the step sequence $l_{c1} \ldots l_{cT}$ 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^2_{ct}, remain in a continuous space.

In the next section, we show how to define the prior distribution of parameters Θ. Section 2.3 details the EM algorithm used to estimate these parameters in maximizing Expression (3).

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. 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.

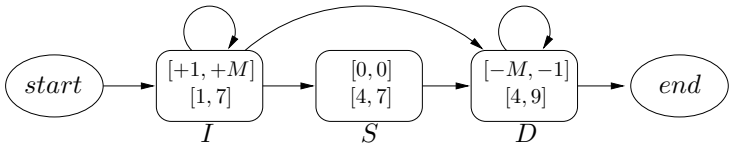

Figure 1. *An HPM for clustering 9-time expression series. In each state, upper and lower intervals represent the step-difference and time intervals associated with the state, respectively. For example, state I allows increasing expression levels, and can be reached between time 1 and 7. This HPM induces bell curve shapes*

2.2.1. *Hidden Phase Models*

The general assumption behind HPMs is that the genes of a given component pass through *phases* or *biological states* over 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. Note that if $s \in Out(s)$ then there is a loop on state 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.

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:

1) t included in the time interval $\tau(s_t)$;

2) $\forall t \geq 2$, $(l_t - l_{t-1})$ 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, $start - I - I - I - I - S - D - D - D - D - end$. For the step sequence on the bottom, 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.

2.2.2. *Probabilistic HPMs*

Non probabilistic HPMs can be used to express strong constraints that describe an expected profile. For more complex knowledge, and when we do not have any information about profiles and just want to express the fact that we are dealing with time 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:

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

One example of probabilistic HPM is depicted in Figure 2.

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

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

How can probabilistic HMPs be used to take time dependences into account? Informally speaking, taking these dependences into account 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

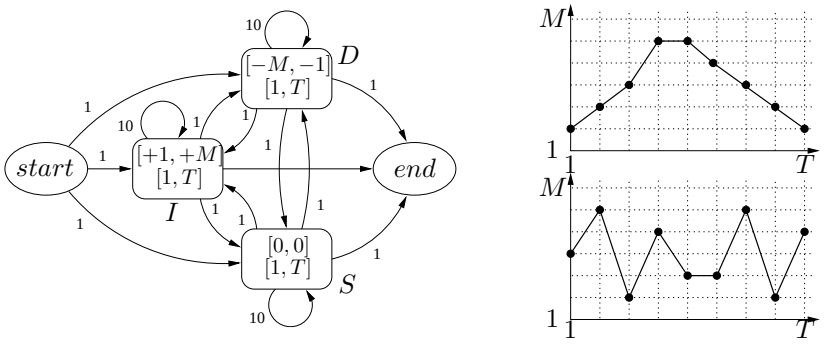

Figure 2. *Left, a probabilistic HPM for clustering expression series without prior knowledge about the form of the profiles. Right, two examples of step sequences*

time, the probability of staying in the same state is higher than the probability of leaving (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. Of course many variants and refinements are possible by changing the weight associated with transitions, or using additional states.

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 (Rabiner, 1989), and is denoted as $V^L = v_0^L \dots v_{T+1}^L$. For example, the Viterbi sequences —and, in this example, sole compatible 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.

2.2.3. *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 $\sum_{L \in \mathcal{L}_T} P(L|H_c) = 1$, 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 $\sum_{L \in \mathcal{L}_T} P(L|H_c) = 1$. For example, for the HPM of Figure 2, the prior probabilities of the two step sequences are proportional to $1/3 \cdot 10/13 \cdot 10/13 \cdot 10/13 \cdot 1/13 \cdot 1/13 \cdot 10/13 \cdot 10/13 \cdot 10/13 \cdot 1/13 \sim 3 \cdot 10^{-5}$ and $1/3 \cdot 10/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \cdot 1/13 \sim 3 \cdot 10^{-10}$, 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 describe the learning algorithm used to estimate parameters Θ of the mixture model. 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 $\mathcal{C} = \{c_X \in \{1, \ldots, C\}, \forall X \in \mathcal{X}\}$ is known, the complete-data likelihood can be written as

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

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:

$$\begin{aligned} Q(\Theta, \Theta^{(i)}) &= E\left[\log P(\mathcal{X}, \mathcal{C}|\Theta)|\mathcal{X}, \Theta^{(i)}\right] \\ &= \sum_{\mathcal{C}\in\mathbf{C}} \log P(\mathcal{X}, \mathcal{C}|\Theta) P(\mathcal{C}|\mathcal{X}, \Theta^{(i)}), \end{aligned}$$

with **C** being the space of values $\mathcal{C}$ can take.(Dempster *et al.*, 1977) show that one can maximize Expression (3) by searching for, at each step of the algorithm, the parameters that maximize the quantity

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

After some calculus —see for example (Bilmes, 1997) for details— this expression can be rewritten as

$$\begin{aligned} Q(\Theta, \Theta^{(i)}) + \log P(\Theta) = \sum_{c=1}^{C} \sum_{X\in\mathcal{X}} \log \pi_c \; P(c|X, \Theta^{(i)}) + \\ \sum_{c=1}^{C} \sum_{X\in\mathcal{X}} \log P(X|c, \Theta) \; P(c|X, \Theta^{(i)}) + \log P(\Theta). \end{aligned}$$

Integrating Expressions (8), (7) and (5), and noting that K'_c are independent of Θ, maximizing the above expression involves maximizing

$$\begin{aligned} \sum_{c=1}^{C} \sum_{X\in\mathcal{X}} \log \pi_c P(c|X, \Theta^{(i)}) + \\ \sum_{c=1}^{C} \left[\sum_{t=1}^{T} \sum_{X\in\mathcal{X}} \log P(x_t|l_{ct}, \sigma^2_{ct}) P(c|X, \Theta^{(i)}) + \right. \\ \left. \sum_{t=1}^{T+1} \log P(v_t^{L_c}|v_{t-1}^{L_c}, t) \right], \quad [10] \end{aligned}$$

with $v_0^{L_c} \ldots v_{T+1}^{L_c}$ the Viterbi sequence of the step sequence L_c in the HPM H_c. Since the two terms of this expression are not related, the new π_c can be estimated by maximizing the first term. Using Lagrange multiplier we get:

$$\pi_c^* = \frac{1}{|\mathcal{X}|} \sum_{X\in\mathcal{X}} P(c|X, \Theta^{(i)}). \quad [11]$$

For the θ_c, we maximize the second term of Expression (10). This involves independently maximizing, for each component c, the quantity

$$\sum_{X\in\mathcal{X}}\sum_{t=1}^{T}\log P(x_t|l_{ct},\sigma^2_{ct})P(c|X,\Theta^{(i)})+\sum_{t=1}^{T}\log P(v_t^{L_c}|v_{t-1}^{L_c},t) \qquad [12]$$

Note that, due to the independence assumptions between times, for each time t, given a step l, one can easily compute the σ^2_{ct} that maximizes the above quantity: it is simply the σ^2_{ct}, denoted as σ^2_{ctl}, that maximizes the sum

$$\sum_{X\in\mathcal{X}}\log P(x_t|l,\sigma^2_{ct})P(c|X,\Theta^{(i)}).$$

Taking the derivative of this expression with respect to σ^2_{ct} and setting it at zero, we get:

$$\sigma^2_{ctl}=\frac{\sum_{X\in\mathcal{X}}(x_t-m(l))^2P(c|X,\Theta^{(i)})}{\sum_{X\in\mathcal{X}}P(c|X,\Theta^{(i)})}. \qquad [13]$$

For L_c the situation is quite different since it is involved in the expression of $P(\Theta)$. The L_c that maximizes Expression (10) 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 (10). This involves searching for the step sequence $L^*_c=l^*_{c1}\dots l^*_{cT}$ that maximizes

$$Z_c(L=l_1\dots l_T)=\sum_{t=1}^{T}\Big[\log P(v_t^L|v_{t-1}^L,t)+\sum_{X\in\mathcal{X}}\log P(x_t|l_t,\sigma^2_{ctl_t})P(c|X,\Theta^{(i)})\Big], \qquad [14]$$

with $\sigma^2_{ctl_t}$ computed with Formula (13). However, as the total number of length T sequences is equal to M^T, enumerating them all is clearly not suitable. We give here a polynomial solution to this problem. It is a dynamic programming algorithm that makes use of the variable $\delta_{cj}(l,s)$, defined as the best score $Z_c(l_1\dots l_j)$ —related to expression (14)— that can be achieved with a step sequence of length j that ends on step l and state s at time j :

$$\delta_{cj}(l,s)=\max_{\substack{l_1\dots l_j\in\mathcal{L}_j\ |\ l_j=l\\ \text{and }(l_j-l_{j-1})\in\epsilon(s)}}\ \max_{s_0\dots s_j=s}\sum_{t=1}^{j}\Big[\log P(s_t|s_{t-1},t)+\sum_{X\in\mathcal{X}}P(c|X,\Theta^{(i)})\log P(x_t|l_t,\sigma^2_{cl_tt})\Big].$$

By induction we have

$$\delta_{cj+1}(l,s) = \max_{\substack{1\leq l'\leq M \mid \\ (l-l')\in\epsilon(s)}} \max_{s'\in\mathcal{S}} \delta_{cj}(l',s') + \log P(s|s',j+1) + \sum_{X\in\mathcal{X}} P(c|X,\Theta^{(i)}) \log P(x_{j+1}|c,l,\sigma^2_{clj+1}). \quad [15]$$

With Formula (15) we can iteratively compute, from $t=1$ to $t=T$, the best score $Z(l_1 \dots l_T)$ that can be achieved with a length T sequence. Moreover, in order to be able to retrieve, at the end of the iterations, the sequence that actually achieves this score, one has to keep track of the step l' and state s' that maximize (15) at each step of the process. This is done by way of the variables $\Psi_{cj}(l,s)$ and $\Phi_{cj}(l,s)$. One can then write the complete algorithm (see algorithm 1).

Algorithm 1: Research of the optimal step sequence

foreach step l **do**
  compute σ^2_{cl1} with Formula (13)
  foreach state s **do**
    $\delta_{c1}(l,s) = \log P(s|start,1) + \sum_{X\in\mathcal{X}} P(c|X,\Theta^{(i)}) \log P(x_1|c,l,\sigma^2_{cl1})$

for $t=2$ to T **do**
  foreach step l **do**
    compute σ^2_{clt} with Formula (13)
    foreach and state s **do**
      $\delta_{ct}(l,s) = \max_{\substack{1\leq l'\leq M \mid \\ (l-l')\in\epsilon(s)}} \max_{s'\in\mathcal{S}} \delta_{ct-1}(l',s') + \log P(s|s',t) + \sum_{X\in\mathcal{X}} P(c|X,\Theta^{(i)}) \log P(x_t|c,l,\sigma^2_{clt})$
      $\Psi_{ct}(l,s) = \operatorname{argmax}_{\substack{1\leq l'\leq M \mid \\ (l-l')\in\epsilon(s)}} \max_{s'\in\mathcal{S}} \delta_{ct-1}(l',s') + \log P(s|s',t) + \sum_{X\in\mathcal{X}} P(c|X,\Theta^{(i)}) \log P(x_t|c,l,\sigma^2_{clt})$
      $\Phi_{ct}(l,s) = \operatorname{argmax}_{s'\in\mathcal{S}} \max_{\substack{1\leq l'\leq M \mid \\ (l-l')\in\epsilon(s)}} \delta_{ct-1}(l',s') + \log P(s|s',t) + \sum_{X\in\mathcal{X}} P(c|X,\Theta^{(i)}) \log P(x_t|c,l,\sigma^2_{clt})$

$l^*_{cT} = \operatorname{argmax}_{1\leq l\leq M} \max_{s\in\mathcal{S}} \delta_{cT}(l,s)$

$s^*_{cT} = \operatorname{argmax}_{s\in\mathcal{S}} \max_{1\leq l\leq M} \delta_{cT}(l,s)$

for $t=T-1$ to 1 **do**
  $l^*_{ct} = \Psi_{ct+1}(l^*_{ct+1}, s^*_{ct+1})$
  $s^*_{ct} = \Phi_{ct+1}(l^*_{ct+1}, s^*_{ct+1})$

The loop of line 1 initializes variables $\delta_{ct}(l, s)$ at time 1. The loop of line 2 is the recursion step. Lines 3 and 4 search for the step and the state that maximize the score at time T, in order to initiate the backtracking step of loop 5.

Let us study the time complexity of this algorithm. The initialization step takes $O(MRN)$ computing time, with M, R and N being the number of steps of the discretized mean space, the number of states of the HPM and the number of series in the data, respectively. In the recursion step, $O(MRN)$ operations are done at each time and for each step-state pair. The total time complexity of this step is then $O(TM^2R^2N)$. Lines 3 and 4 take $O(MR)$ computing time, and the backtracking is in $O(T)$. The total time complexity of the algorithm is then $O(TM^2R^2N)$.

Algorithm 2: Learning algorithm

```
Set parameters to initial values
repeat
    for c = 1 to C do
        compute π*_c with Formula (11)
        foreach time t and step l do
            compute σ²_ctl with Formula (13)
        Find the optimal step sequence L*_c = l*_c1 ... l*_cT with the dynamic
        programming algorithm
    Compute P(X|Θ)P(Θ)
until convergence
```

One can now write the complete learning algorithm —see Algorithm 2. 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, and the algorithm converges toward a local optimum. 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. The time complexity of the the complete learning procedure is $O(BCTM^2R^2N)$, with B being the maximal number of loops of the EM algorithm.

3. Evaluation and experiments

When applied to a given dataset, our method provides a mixture model, i.e. a set of profiles (the step sequences) with the variances σ^2_{ct} associated with each time and the prior probabilities π_c. Moreover, it provides the probability membership of each gene for each class, and groups the genes according to their most probable components into clusters. Features of the mixture model are useful to access the pertinence of the clusters. Indeed, sometimes one constrained component c may fail to collect "good" genes. This occurs when no gene agrees with H_c, or when the desired genes are collected by another component with similar constraints. Then, two different situations can arise. First, the component does not collect any gene and its probability

$\pi_c = 0$. Second, the component collects some series, but these do not have the desired profile: the measures x_t are far from $m(l_{ct})$ at one or several time points (there is a gap between the series and L_c) and the variance is high at these points. This situation can be merely detected by visual inspection, or by checking if the value of σ^2_{ct} is not higher than a given threshold.

3.1. *Recovering a known class of genes*

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 (see Section 3.3 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 3 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 3 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 4.

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.

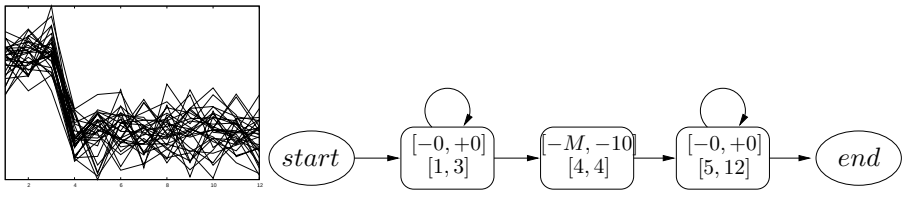

Figure 3. *Left, examples of synthetic expression series added to the fibroblast dataset. Right, the HPM designed to find the synthetic class among the "real" biological classes in the fibroblast dataset*

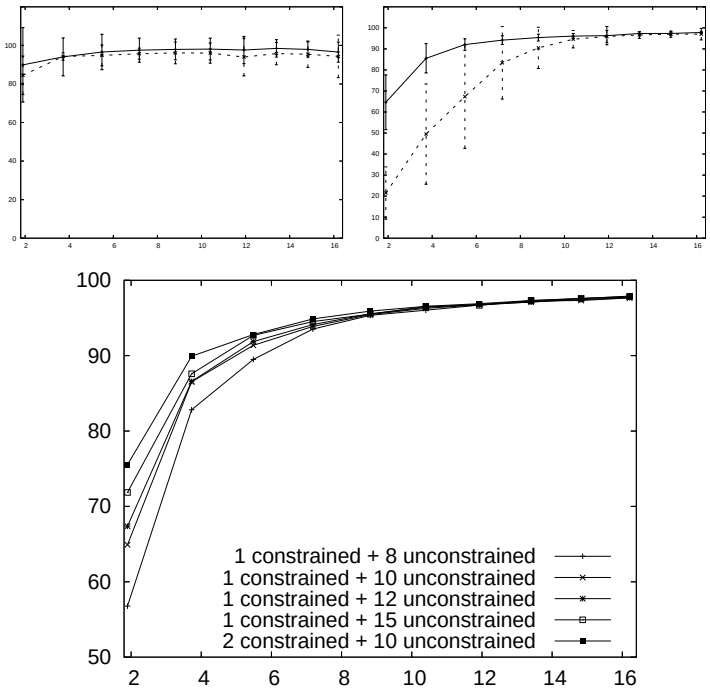

Figure 4. *Up: recall (left) and precision (middle) achieved with (solid lines) and without (dashed lines) prior knowledge about the class of interest. The x-axes denote the proportion (in percent) of this class among all the expression series. Bottom: precision achieved using different number of components*

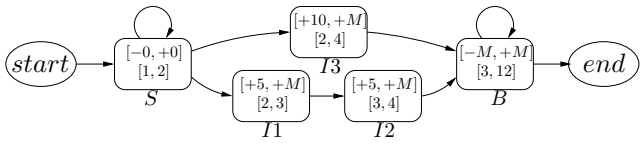

Figure 5. *An HPM to uncover quick over-expression classes*

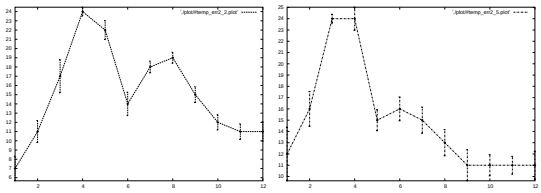

Figure 6. *Fibroblast dataset. Two profiles obtained with the help of the HPM of Figure 5*

3.2. *Number of components*

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 —see bottom of 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.

3.3. *Fibroblast dataset*

Next, some experiments to find "real" classes in the Fibroblast dataset have been carried out. This is the dataset of (Iyer *et al.*, 1999). Authors study the response of human fibroblasts to serum. The expression level of 8613 genes have been measured at 12 times, ranging from 15 min to 24 hours after serum stimulation. The authors selected a subset of 517 genes whose expression changed substantially in response to serum. The same subset, centered and reduced on genes is used here. First we

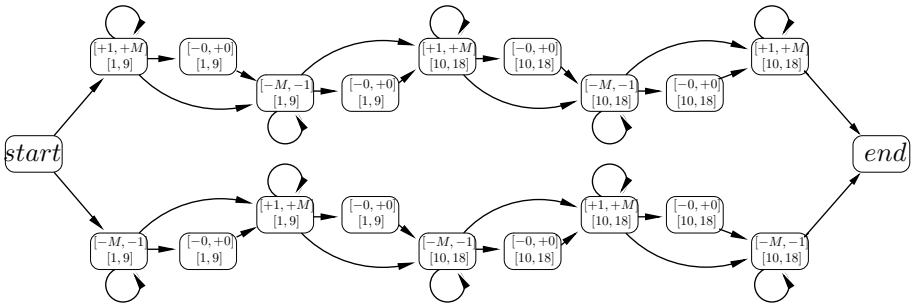

Figure 7. *An HPM to find out sinusoidal profiles*

clusterized the original data in 10 classes, using only knowledge about their temporal feature, i.e., by constraining all the components with an HPM like that of Figure 2. This clustering leads to 10 profiles relatively similar to those that (Iyer *et al.*, 1999) defined by hand after a hierarchical clustering. While most of these classes are well-defined, within them it is hard to identify genes that show a quick response to the serum. Only one jumbled class seems to present this feature. We designed an HPM specially adapted to such class (see Figure 5). The state S of this HPM models a potential and short —until time 2, at maximum— delay phase before over-expression. Next, 3 states are used to model the increasing phase: this can be quite moderate (at least 5 steps) during 2 times at least (states $I1$ and $I2$), or heavy (at least 10 steps) during 1 time (state $I3$). In both cases, the aim is to observe marked over-expression before time 5. The last state models the remainder of the class and is not constrained —all increases and decreases are allowed. We use a 10 components mixture model, with 3 components constrained with this special HPM, and 7 components constrained with an HPM like that of Figure 2.

Classes with the desired profile have been uncovered by this method. Figure 6 shows the mean profile of two classes. The third class has a very high variance at time 2, and a visual inspection shows that the collected series actually diverge from the profile at this point, so the class is not interesting. The two classes of Figure 6 differ by the time when genes reach their maximal over-expression —times 3-4 and times 4-5. Note that these classes show a second increase step which is not specified in the HPM we used. This illustrates the ability of the method to uncover the desired classes even when their profiles are not completely specified.

3.4. *Yeast dataset*

This is the dataset published in (Spellman *et al.*, 1998). Authors measure the expression level of 6178 genes 18 times during slightly more than two full cell cycles. We use the same normalization method as in (Spellman *et al.*, 1998): the logarithms

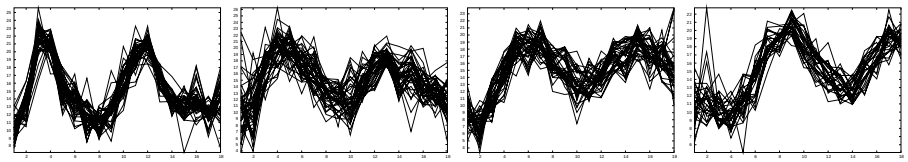

Figure 8. *Yeast dataset. Four classes uncovered with the help of the HPM of Figure 7*

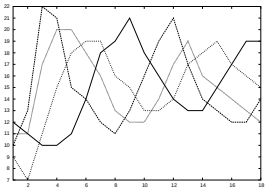

Figure 9. *Yeast dataset. Superimposition of the four mean profiles of the classes of Figure 8*

of the data are centered and reduced on the genes, and genes that do not show any time points higher than 2 or lower than -2 are removed. This leads to a dataset of 1044 expression series. The main aim of the study was to find out cycle-regulated genes. So we look for classes showing a two-time repeat of the same pattern (since series span two cell cycles), i.e., classes with sinusoidal shape. The HPM of Figure 7 is designed for this purpose. It detects profiles that show whether (upper part of the HPM) 2 concave patterns —a concave pattern being an increase followed by a decreasing phase— eventually with a third increasing phase, or (lower part of the HPM) 2 convex patterns eventually followed by a third decreasing phase. Each increase or decrease can be followed by a short (one time) stability phase, and the time constraints of the τ functions require the convex or concave patterns to be equally distributed between first nine and last nine times.

A 20 component mixture model has been used for the clustering. The 10 first components have been constrained with HPM of Figure 7, while the 10 other components were not constrained to sinusoidal profiles but by the probabilistic HPM of Figure 2. Many classes that seem to be regulated by the cell cycle have been uncovered in this way. Figure 8 shows four of theses classes. These four differ by the times genes are over- or under-expressed. When superimposing the mean profiles of these classes on the same graph (see Figure 9), shifts between the different minima and maxima achieved can be seen.

4. Conclusions

We proposed a Bayesian approach for the clustering of short time series. This approach can be used to integrate prior knowledge about the general profile of the classes of interest, or to deal with the temporal nature of the data when no prior knowledge is available. It makes use of a new type of model close to HMMs that we call a Hidden Phase Model. A mixture model is used to model the series, and each component of the mixture is associated with a given HPM. This defines the prior probability distribution of the parameters of the class. Then an EM algorithm is used to estimate the parameters of the mixture in maximizing its posterior probability.

Applied to two different datasets —(Spellman *et al.*, 1998) et (Iyer *et al.*, 1999)—, our method shows good performance and ability to efficiently uncover classes of genes with the desired profiles. In practice, appropriate HPMs can be designed easily and naturally. 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 respect to the total number of series, and that it can be really interesting when this number is very small.

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, Olivier Gascuel and the four anonymous referees for their help and comments on this work.

5. References

Alon U., Barkai N., Notterman D. A., Gish K., Ybarra S., Mack D., Levine A. J., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays", *Proc Natl Acad Sci USA*, vol. 96, num. 12, pp. 6745-6750, Jun, 1999.

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*, vol. 10, num. 3-4, pp. 341-356, 2003.

Bilmes J., A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, Technical Report num. ICSI-TR-97-021, University of Berkeley, 1997.

Dempster A. P., Laird N. M., Rubin D. B., "Maximum likelihood from incomplete data via the EM algorithm.", *J. Royal Stat. Soc. B*, vol. 39, pp. 1-38, 1977.

Duda R., Hart P., Stork D., *Pattern Classification*, John Wiley, 2001.

Eisen M. B., Spellman P. T., Brown P. O., Botstein D., "Cluster analysis and display of genome-wide expression patterns", *Proc Natl Acad Sci USA*, vol. 95, num. 25, pp. 14863-14868, Dec 8, 1998.

Ernst J., Nau G., Bar-Joseph Z., "Clustering short time series gene expression data", *Bioinformatics*, vol. 21 Suppl 1, pp. i159-i168, Jun 1, 2005.

Hertzberg M., Aspeborg H., Schrader J., Andersson A., Erlandsson R., Blomqvist K., Bhalerao R., Uhlen M., Teeri T., Lundeberg J., Sundberg B., Nilsson P., Sandberg G., "A transcriptional roadmap to wood formation", *Proc Natl Acad Sci USA*, vol. 98, num. 25, pp. 14732-14737, 2001.

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*, vol. 283, num. 5398, pp. 83-87, Jan, 1999.

Kohonen T., *Self-Organizing Maps*, Springer, 1997.

Lloyd S., "Least squares quantization in PCM", *IEEE Trans. Info. Theory*, vol. IT-2, pp. 129-137, 1982.

McLachlan G., Krishnan T., *Finite mixture models*, John Wiley, 2000.

Rabiner L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, vol. 77, num. 2, pp. 257-285, 1989.

Ramoni M. F., Sebastiani P., Kohane I. S., "Cluster analysis of gene expression dynamics", *Proc Natl Acad Sci USA*, vol. 99, num. 14, pp. 9121-9126, Jul, 2002.

Schliep A., Schonhuth A., Steinhoff C., "Using hidden Markov models to analyze gene expression time course data", *Bioinformatics*, vol. 19 Suppl 1, num. 14, pp. 255-263, Jul, 2003.

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*, vol. 9, num. 12, pp. 3273-3297, Dec, 1998.

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 USA*, vol. 96, num. 6, pp. 2907-2912, Mar 16, 1999.