The General Hidden Markov Model Library: Analyzing Systems with Unobservable States

Alexander Schliep[1] Wasinee Rungsarityotin[1] Benjamin Georgi[1]
Alexander Schönhuth[2]

[1] Max Planck Institute for Molecular Genetics, Berlin
[2] ZAIK, University of Cologne

Abstract

Hidden Markov Models (HMM) are a class of statistical models which are widely used in a broad variety of disciplines for problems as diverse as understanding speech to finding genes which are implicated in causing cancer. Adaption for different problems is done by designing the models and, if necessary, extending the formalism. The General Hidden Markov Model (GHMM) C-library provides production-quality implementations of basic and advanced aspects of HMMs. The architecture is build around the software library, adding wrappers for using the library interactively from the languages Python and R and applications with graphical user interfaces for specific analysis and modeling tasks. We have found, that the GHMM can drastically reduce the effort for tackling novel research questions. We focus on the Graphical Query Language (GQL) application for analyzing experiments which measure the expression (or mRNA) levels of many genes simultaneously over time. Our approach, combining HMMs in a statistical mixture model, using partially supervised learning as the paradigm for training results in a highly effective, robust analysis tool for finding groups of genes sharing the same pattern of expression over time, even in the presence of high levels of noise.

1 Introduction

When we set out to analyze experimental mass data we have to ask ourselves what is the nature of the underlying system generating the data — what is the physical, chemical or biological process we are investigating and what is the, usually, limited and imperfect view provided by our experimental instruments. Often the processes will be stochastic; moreover, more often than not the experimental procedure will introduce another source of stochasticity.

Provided that they are suffi ciently complex, the systems we are investigating will share one commonality: the observations will be influenced by the state the system is in. Here state very broadly encompasses the values of all variables which describe he system. However, the state itself will be unobservable. This structure can for example be found in understanding speech, where words correspond to states and the waveforms of the spoken word are observed, or in fi nding genes in a DNA sequence, where the various structural elements of a

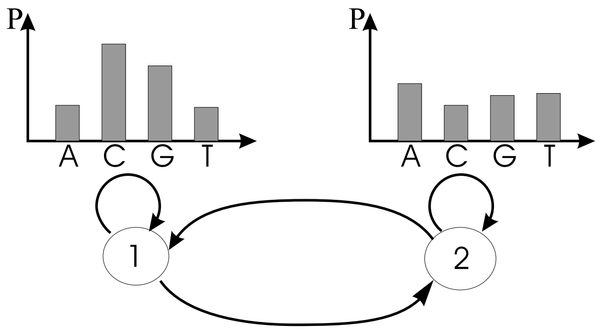


Figure 1: A discrete HMM. The underlying Markov process is depicted as a directed, weighted graph, the states correspond to vertices and transitions to edges. The emissions in each state are displayed as discrete distributions over the alphabet of the four nucleotides. This model can be used to analyze segmentation of DNA based on nucleotide usage differences between the segments.

gene (e.g., exons, introns, splice sites, start and stop codons) are the states and we observe the sequence of nucleotides.

For a multitude of analysis tasks — finding the most likely state given observations, classifying observations into different groups — a stochastic model offering effective computation of such processes with hidden states would be beneficial. If we make the assumptions, that

- the observations only depend on the state the system is in at the time of the observation and thus are independent on prior observations as well as prior states, and
- the probability of a change from a state at time t to another state at time $t + 1$ only depends on the state at time t

then the so-called Hidden Markov Models (HMM) provide an effective model class. In essence they combine two stochastic processes. Consider a stochastic process on the discrete and finite set of states S (extensions to continuous state spaces are routine), a sequence of random variables

$$\{X_t\}_{t\geq 1}, \;\; X_t \in S$$

such that the so called Markov property holds:

$$P[X_{t+1} = s_{t+1}|X_t = s_t, \ldots, X_1 = s_1] = P[X_{t+1} = s_{t+1}|X_t = s_t].$$

If we have a stochastic function $f : S \mapsto R(\Sigma)$, where $R(\Sigma)$ denotes the set of random variables with outcomes Σ, such that for all $i \in \{1, \ldots, N\}$, $f(i)$ is a random variable taking on values from Σ then

$$\{Y_t\}_{t\geq 1}, \;\; Y_t := f(X_t)$$

is a HMM.

Hidden Markov Models in fact allow a wide range of variations with respect to emissions, they can be discrete, continuous, or vector-valued, the densities controlling the random variables f — for continuous emissions mixtures of Gaussian are routinely used — and the details of the Markov process controlling the sequence of states.

The first publications on HMMs stem from the 1940s, but they have not found widespread use until the 1970s when their effectiveness in modeling speech became obvious in the push to implement speaker-independent speech recognition at AT&T. Today HMMs form the basis for a wide range of solutions for data analysis and statistical modeling, from areas such as guiding missiles [18], predicting crises in the Middle East [10] or finding genes in human DNA sequence [7, 22]. While many applications can be addressed with standard HMMs, often extensions to the basic method are required.

Our main contribution is two-fold. On one hand we have implemented our software in the highly reusable, general library GHMM— licensed under the Library GNU Public License (LGPL). We implemented the standard algorithms for computing with HMMs and a large number of extensions, both to the model class and algorithms. So-called wrappers allow the use of GHMM from interactive languages such as Python and R and a graphical application is provided for editing HMMs. The GHMM thus creates a comprehensive, flexible framework which substantially reduces the effort for implementing novel data analysis and modeling solutions. The architecture allows interactive use, incorporation into other software package, and — due to the licensing chosen — extension of the core functionality. Altogether, these aspects lead to a considerable speed-up of research efforts using HMMs.

On the other hand we employ the HMM framework and the GHMM in particular to design a novel mixture of HMMs which performs very well for identifying groups of genes in gene expression time-courses, due to the ability to use more prior knowledge than competing approaches and a large degree of flexibility in modeling the qualitative behavior of time courses.

In the following we will describe out work on analysis of gene expression time-courses in detail, list further application of the GHMM, and expand about the architecture of the GHMM.

2 Analysis of gene expression time course data using mixtures of Hidden Markov Models

Microarray experiments have become a staple in the experimental repertoire of molecular genetics. They can be used to detect or even quantify the presence of specific pieces of RNA in a sample. The experimental procedure is based on hybridization of these RNA-sequences to either oligonucleotide or cDNA probes which are affixed to the array. If probes for genes in a cell are used, microarray experiments can measure the expression levels of up to thousands of genes simultaneously. The resulting so-called expression profiles allow for example investigation of differences in distinct tissue types or between healthy or diseased tissues. When microarray experiments are performed consecutively in time we call this experimental setting a time course of gene expression profiles. The questions this experimental setting tries to address are the detection of the cellular processes underlying the regulatory effects observed, inference of regulatory networks and, in the end, assigning function to the genes analyzed in the time courses.

Because of the large number of genes and their complex relationships in microarray measurements, it has been a standard procedure to identify groups of genes with similar temporal regulatory patterns or *time-courses*. When analyzing such gene expression time-courses a number of problems should be addressed.

- Noise is omnipresent and of manifold nature. We need a good statistical model to deal with it.

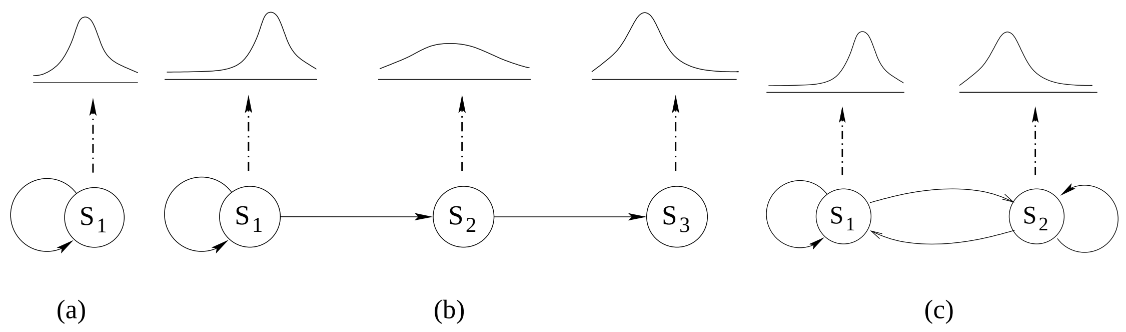


Figure 2: A number of prototype HMMs encoding distinct qualitative time-course behavior: constant (a), up-down-up (b), cyclic up-down (c).

- Sometimes prior knowledge in form of high quality annotations regarding regulation or function of the inspected genes is available. The method should thus allow for the integration of readily labeled data.
- The computer aided analysis of gene expression experiments is an experimental help for the biologist. This means that the tool should allow for a high degree of interactivity and visualization.
- Very often only a small number of genes is decisively involved in the processes of interest. The procedure should thus be able to output only few genes having highly significant relationships if required.
- Genes can trigger the expression of other genes. So gene expression profiles which are to be grouped together may exhibit similar expression patterns showing up at different times.
- Sometimes experiments are (partially) corrupted leading to missing data.
- Genes may interact with other genes in more than only one context. This precludes partitioning the data.
- Gene expression profiles are the result of a time course experiment. Methods which take care of these so called *horizontal dependencies* should outperform those which do not.

Prior approaches can be divided into two classes, depending on whether they are based on statistical models or not. Methods in the second class require the definition of a distance measure describing the degree of similarity between two gene expression profiles. Note that these methods mostly do not explicitly account for the high levels of noise in the data. Moreover they do not allow for the inherent nature of the data, namely being time courses. Examples are hierarchical [5, 6] and k-means clustering [19] or singular value decomposition [14]. Some of them provide graphical user interfaces, some do not. None of these methods allows for integration of prior knowledge.

Methods of the first class use statistical models to represent clusters. Cluster membership is decided based on maximizing the likelihood of data points given the cluster models and the assignment of data points to clusters. Model based procedures account for the horizontal dependencies in the data. Moreover one can expect a larger robustness with respect to noise as it mostly is explicitly modeled in these approaches. Examples of model-based clustering used for analysis of expression time courses are based on cubic splines [1] and autoregressive curves [12, 13].

2.1 Mixtures of Hidden Markov Models

To cope with the issues given, we model a set of gene expression time-courses as a mixture model. The basic assumption of a mixture model is that the data has been generated by a weighted superposition of model components coming from the same model class but differing in their parameters and their weights. Besides from providing a “soft” assignment of time courses to clusters mixture models also have proved to be more robust with respect to noise when learned from data. The individual components we use are HMMs, mainly due to their flexibility in encoding ’grammatical’ constraints of time-courses. Their graphical structure benefi ts the analysis process, as it affords a high degree of interactivity and accessibility.

Simple model for time-courses

We use HMMs (see Rabiner [11] for an excellent introduction) with continuous emissions governed by a normal distribution in each state. The HMM topology — the number of states, the set of possible transitions — is essentially a linear chain (following Schliep et al. [15], see Fig. 2), neglecting a possible transition from the last to the fi rst state to accommodate cyclic behavior. The states reflect regions of a time-course with *similar* levels of expression. There are usually fewer states than time-points, as several similar successive measurements will be accounted for by the same state by making use of its self-transition. It is important to point out that our approach is not limited to such models but rather accommodates arbitrary HMM topologies.

We deal with missing values in the following way. Each state of an HMM can either emit a real-valued variate according to its Gaussian state emission pdf or, with a low probability equal to the proportion of missing values in all the time-courses, a special missing symbol.

Learning Mixtures

We combine K of such HMMs $\lambda_1, \ldots, \lambda_K$ to a probability density function (pdf) for a gene expression time-course by use of a convex combination of the K component probability density functions induced by the HMMs, denoted $p_j(\cdot, \lambda_j)$. The mixture pdf is parameterized by $\Theta = (\lambda_1, \ldots, \lambda_K, (\alpha_1, \ldots, \alpha_K))$ and defi ned as

$$p(\cdot|\Theta) := \sum_{j=1}^{K} \alpha_j p_j(\cdot, \lambda_j).$$

As the former is just a usual mixture [8, 9], the well-known theory applies. The resulting likelihood function can be optimized with the EM-algorithm [2, 3, 4, 21].

We additionally propose to use *labeled* data by extending the EM algorithm to gain from prior knowledge. We show that there is a large improvement in convergence to *good* local optima on typical data, even if only small amounts of labeled data are supplied.

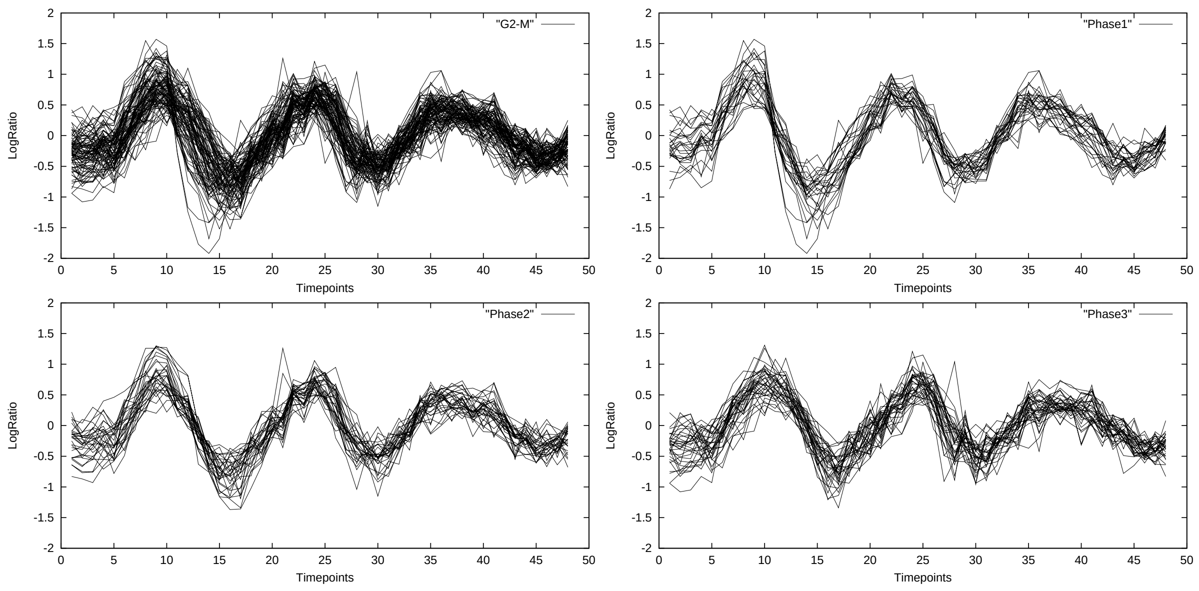


Figure 3: A group obtained by computing a mixture model using nine labeled and 2263 unlabeled time-courses from the Whitfi eld data set (top left). It contains fi ve of the labeled time courses. The group was decomposed, using the Viterbi decomposition, into three subgroups, corresponding to synchronous genes, resulting in a fi rst subgroup containing mainly G2 genes (bottom left, phase 1), the second having G2 as well as G2/M genes (top right, phase 2) and the third having mostly G2/M genes (bottom right, phase 3).

To apply the EM-algorithm one assumes the existence of unobservable (or hidden) data $Y = \{y_i\}$, which indicates which component has produced each O^i in the set of time-courses $\mathcal{O}$. Thus, we can formulate a complete-data log-likelihood function $\log L(\Theta|\mathcal{O}, Y)$.

If we are given additional labeled time-courses, we do not have to guess the corresponding y_i. We denote the set of labeled time-courses with $\mathcal{O}_L$ and the set of unlabeled ones with $\mathcal{O}_U$. For a time-course O^i from $\mathcal{O}_L$ we set the value of y_i to its component label l_i and maintain this assignment throughout the running time by setting $\mathbb{P}[\lambda_j|O^i] = 1$ for $j = l_i$ and zero else. The Θ^t are the estimates for the maximum likelihood in the t-th iteration), which splits into two sums,

$$Q(\Theta, \Theta^t) := \sum_{O^i \in \mathcal{O}_L} \log\left(\alpha_{l_i} p_{l_i}(O^i|\lambda_{l_i})\right) +$$

$$\textstyle\sum_{O^i \in \mathcal{O}_U} \sum_{j=1}^{K} \log\left(\alpha_j p_j(O^i|\lambda_j)\right) \mathbb{P}[j|\Theta^t, O^i],$$

and for which the usual local convergence result holds.

Inferring Groups

The simplest way of inferring groups in the data, is to interpret the mixture components as clusters and assign each time-course to the cluster which maximizes the probability of the cluster given the time-course O, $\mathbb{P}[\lambda_j|O]$. However, a mixture encodes much more information. Inspection of the discrete distribution $d(O) := \{\mathbb{P}[\lambda_i|O]\}_{1 \le i \le K}$ reveals the level of ambiguity in making the assignment, which can be quantifi ed easily and sensibly by computing the entropy $\mathbb{H}(d(O))$. Choosing a threshold on the entropy yields a grouping of the data

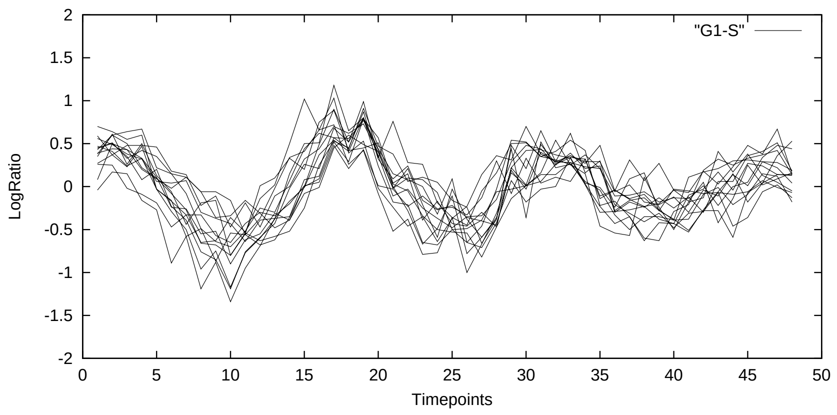


Figure 4: Another group containing cell cycle related genes obtained by computing a mixture model using nine labeled and 2263 unlabeled time-courses from the Whitfi eld dataset. This group contains only genes belonging to phases G1/S and S, four of which were labeled input.

into $K + 1$ groups, one group containing all profi les showing no signifi cant membership to one of the components.

Groups will typically contain time-courses having the same qualitative behavior. The time at which, for example, an up-regulation occurs will often vary. Synchronous subgroups of such clusters are found with the Viterbi-decomposition introduced in [15].

2.2 Results

Biological data

We used published data from a time-course experiment [20], in which the authors measured genome wide gene expression of synchronized HeLa (cervical cancer cells) cells. Goal of the experiment was the detection of genes regulating cell cycle. One cycle can be divided into fi ve phases each of which representing a section of life of a eukaryotic cell between two typical events such as mitosis or division. Genes, which are involved in the regulation of the cell cycle, are further classifi ed according to their regulation levels in different phases. The data was pre-processed by extracting all those genes with an absolute fold change of at least two in at least one time point. This resulted in a data set containing 2272 expression time courses.

The method was run using a collection of 35 random linear 24-state models. We used fi ve G2/M phase genes described above as a seed for one cluster and four genes of the G1/S phase for a second one. We inferred two groups containing the labeled time-courses of size 91 and 14 respectively, see Figs. 3, 4. We computed a Viterbi-decomposition of the larger group thus fi nding three subgroups, one containing only G2/M, the second containing G2/M and G2 genes and the third containing only G2 genes. The second cluster, see Fig. 4, contained twelve G1/S and two S-phase genes. All time-courses that are assigned to the different phases of our G2, G2/M phase cluster are known to be cell cycle regulated in their respective phase [20]. The same holds for the G1/S, S phase cluster. Thus, the modest amount of prior information used resulted in highly specifi c (sub-)groups of synchronously expressed genes.

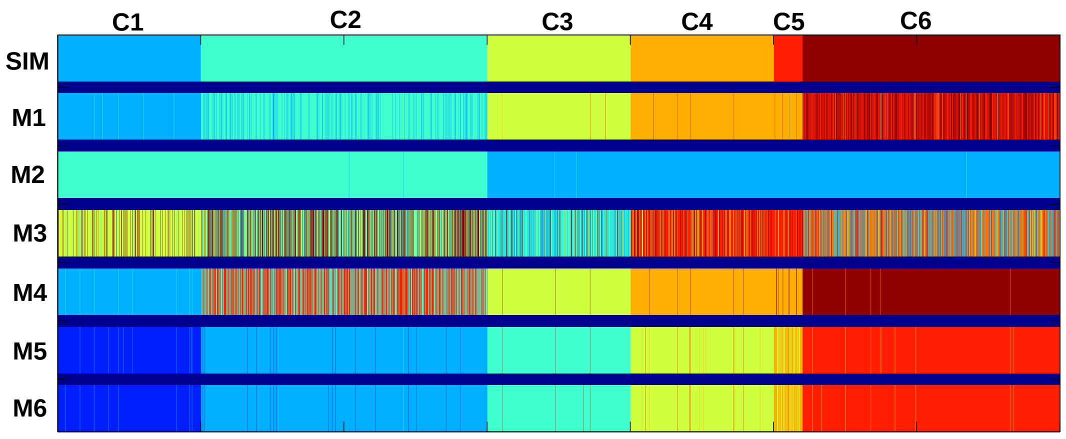


Figure 5: Artificial data allows easy comparison: Cluster assignment of time-courses in `SIM`: The first column shows class C1-C6 from the simulated data set `SIM` as colored blocks, the second to the seventh column shows the cluster assignments obtained by methods M1-M6 (see Table 1 for method descriptions). The class C1 correspond to up-regulated genes, C2 to noise, C3 to down-regulated genes and C4-C6 are cyclic genes (see [16] for details).

Simulated Data

To facilitate benchmarking and evaluation we tried to design a method for creating simulated data sets, which makes very mild assumptions about the nature of the data but reflects the realities of microarray experiments. Our proposed approach is *independent* from the underlying assumptions and peculiarities of the statistical model in our method, as it is independent from the assumptions in other methods.

As shown in Table. 1, two of the more involved methods, Caged [13] and the Spline based clustering by Bar-Joseph et al. [1] only reach a specificity of less than 50%. The main error made by Caged in deciding on too few clusters (this cannot be controlled by the user) which leads to merging of several classes (C1 and C2 respectively C3-C6, cf. Fig. 5) into one cluster. The HMM mixture perform quite well, achieving a high degree of over 90% specificity and over 75% sensitivity. The tests also show very clearly the impressive effect of partially supervised learning. It suffices to have labels for thirty, or less than one percent of all time-courses (cf. M5 in Table. 1), to obtain a specificity and sensitivity exceeding 95%. More labels do not yield further significant improvements.

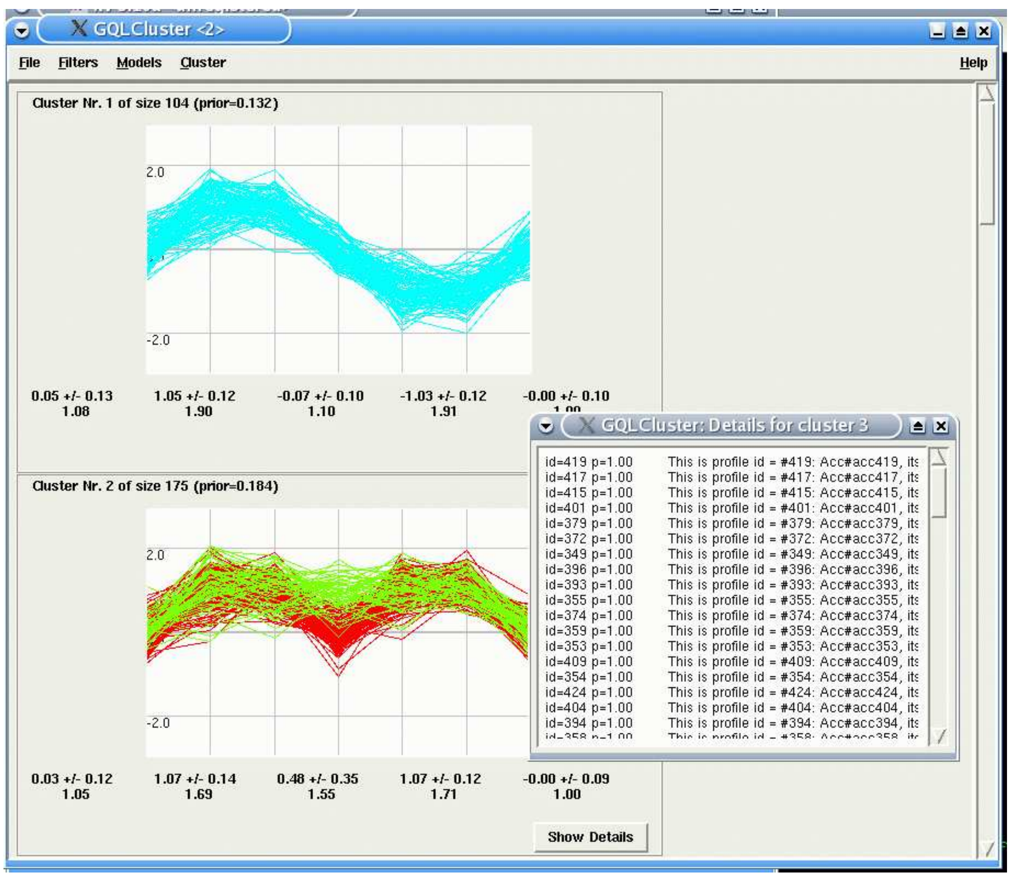
3 Further applications

The GHMM is in use in a wide range of research, thesis and industrial projects. The fields include computational finance (liquidity analysis), physiology (analysis of EEG data), computational linguistics and astronomy (classifying stars). Projects in our group mostly address problems from molecular biology, for example finding genes, assigning function to proteins, and discovering hierarchical groups in protein space.

In the following we will briefly introduce two projects currently under research, which are typical in the sense that they would not have been started without the library supplying most of the necessary functionality.

Table 1: Results on the simulated data set SIM for k-means clustering, CAGED [13], Splines [1], and HMM Mixtures with no, 0.9% (five per class) and 1.7% (ten per class) labeled time-courses per class. By comparing the known classes in SIM with the computed clustering for all pairs of time-courses we computed true and false positives as well as true and false negatives, abbreviated TP, FP, TN and FN. True positive is defined as a pair of time-courses with equal class which are assigned to the same cluster. To quantify the performance we computed the standard sensitivity, $\frac{\#TP}{\#TP+\#FN}$, and specificity, $\frac{\#TP}{\#TP+\#FP}$.

Method	**Description**	**Specificity**	**Sensitivity**
M1	k-means, Euclidean distance	85.55%	71.87%
M2	CAGED	41.00%	99.70%
M3	Splines	47.29%	39.38%
M4	HMM Mixtures	93.00%	79.14%
M5	HMM Mixtures, 0.9% labeled time-courses	96.40%	96.90%
M6	HMM Mixtures, 1.7% labeled time-courses	96.60%	96.99%



```
id=419 p=1.00      This is profile id = #419: Acc#acc419, its
id=417 p=1.00      This is profile id = #417: Acc#acc417, its
id=415 p=1.00      This is profile id = #415: Acc#acc415, its
id=401 p=1.00      This is profile id = #401: Acc#acc401, its
id=379 p=1.00      This is profile id = #379: Acc#acc379, its
id=372 p=1.00      This is profile id = #372: Acc#acc372, its
id=349 p=1.00      This is profile id = #349: Acc#acc349, its
id=396 p=1.00      This is profile id = #396: Acc#acc396, its
id=393 p=1.00      This is profile id = #393: Acc#acc393, its
id=355 p=1.00      This is profile id = #355: Acc#acc355, its
id=374 p=1.00      This is profile id = #374: Acc#acc374, its
id=359 p=1.00      This is profile id = #359: Acc#acc359, its
id=353 p=1.00      This is profile id = #353: Acc#acc353, its
id=409 p=1.00      This is profile id = #409: Acc#acc409, its
id=354 p=1.00      This is profile id = #354: Acc#acc354, its
id=424 p=1.00      This is profile id = #424: Acc#acc424, its
id=404 p=1.00      This is profile id = #404: Acc#acc404, its
id=394 p=1.00      This is profile id = #394: Acc#acc394, its
id=358 p=1.00      This is profile id = #358: Acc#acc358, its
```

Figure 6: The method is implemented in a GUI-application written in Python using the highly portable Tk widget set. The mixture estimation is also written in Python, as the function calling overhead is negligble and all computationally intense work is handled by the GHMM respectively by the Numeric package for Python.

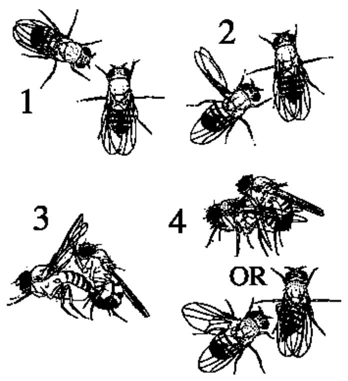


Figure 7: Mating behavior of Drosophila male and female. From a recording of a pair of fruit flies, we can create a training set that contains the annotation of observed behavior as well as mating songs. A male Drosophila is slightly smaller than female one and thus allows a possible semi-unsupervised system of the annotation of the dancing pair by using HMM trained from the manually annotated segments.

3.1 Mating dances of Drosophila

One of the fascinating facts about Drosophila (fruit fly) is that they dance in pairs. Their inherited dances specifi cally serve as a selection criterion during courtship. To study the mating behavior and understand phenotypic effects of genetic mutations Drosophila biologists collect the data [1] by observing a pair of male and female flies over a period of time and annotating their actions with a controlled vocabulary. The manually transcribed poses for wild types and fly mutants form a language of unknown structure. HMMs will be used to detect motifs in the dances and build models for wild types and mutants, predicting mating.

A second, later objective is to replace the manual annotation by an automated procedure. When the recording environment can be constrained to one static camera and a planar view, we can design a semi-supervised system to perform annotation of dances directly. We can then model a sequence of video frames, instead of the annotated dance steps, or dancing poses as an observation sequence produced by an HMM.

3.2 Gene Expression and Chromosomal Proximity

Finding genetic causes for such serious and prevalent diseases as cancer is a very important and very diffi cult task. Usually, there will not be a single locus variation causing the disease but rather a combination of factors. One of the factors are chromosomal aberrations which do change the levels of expression of genes in close positional proximity on the chromosome. The problem is to identify groups of genes from contiguous regions which exhibit the same, possibly weak difference in expression when one compares cells from healthy and diseased samples. Using the positional information in addition to the differential expression should yield superior results.

An HMM with states for same, higher and lower levels of expression when comparing healthy and in diseased tissues is an effective model for the sequence (with respect to position) of observations. The positional effect can be explicitly modeled by using a *non-homogeneous*

[1]Principal investigator is Benjamin Arthur at the ETH.

Markov chain. The probability of seeing a higher level of expression in gene $i+1$, given that gene i shows a higher level should decrease with the distance between genes i and $i+1$ on the chromosome. This extension to standard HMMs is already part of the GHMM and this application is the focus of a research project at our institute [2].

4 Software

At the core of the GHMM, see Fig. 9 is the GHMM C-library which provides efficient implementations of many HMM variants and their relevant algorithms:

- **Observations:** Discrete, continuous, vector valued, ”silent” emissions, observations conditioned on previous observations (higher-order states)
- **Observation densities:** Discrete, uni-variate Normal, truncated uni-variate Normals, mixture of (truncated) uni-variates Normals, multi-variate Normals
- **Markov chain:** Discrete state space, time-homogeneous, time-inhomogeneous (discrete classes)
- **Training:** Expectation-Maximization, Gradient Descent, discriminative learning
- **Probabilities:** Likelihoods, many marginals
- **Decoding:** Viterbi, 1-best, posterior

Wrappers — pieces of code which allow easy, native access to C-libraries from Python, C++ and R — provide one possible interface. This allows to develop applications using the GHMM, such as the GQL, in either language. Moreover, the Python and R wrappers allow interactive use of the GHMM from the command line. The HMMEd editor adds a powerful graphical user interface for the design and modification of HMMs. The different layers are linked by our HMMXML, implementing XML input and output as an ubiquitous format for both models and sequences.

Our design choice assumes a novel type of computational scientist as the user of our software, the “scripter”. In statistics and numerical analysis software packages such as S, R, and

[2] Principal investigator is Stefan Röpcke at the Max Planck Institute for Molecular Genetics

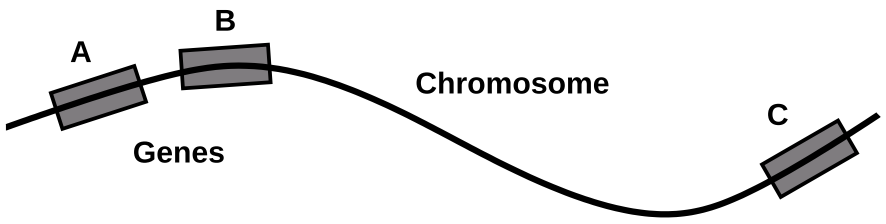


Figure 8: We compare the levels of gene expression for complete chromosomes in healthy and in diseased tissues. The position of the genes on the chromosome are known. If chromosomal aberrations are causing disease, then one would expect a higher chance of observing differences in expressions if the genes are close. For example, genes A and B should exhibit a positional effect, whereas B and C can be treated as independent.

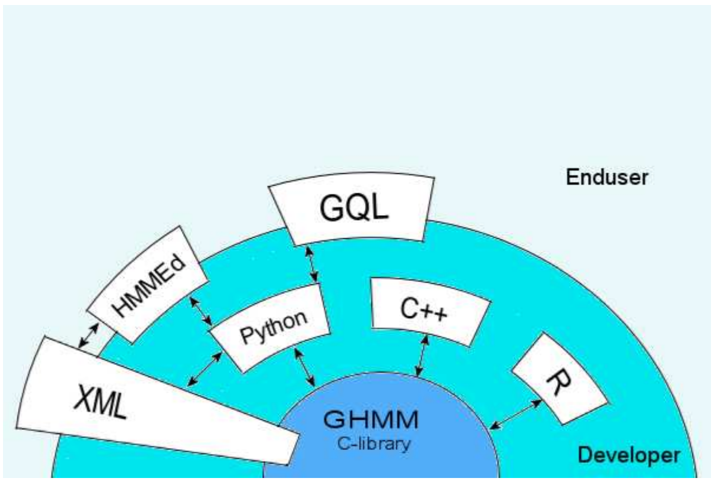


Figure 9: The GHMM core library is written in C. We provide wrappers to use it from C++, Python and R. Additional packages use and extend GHMM through provision of graphical user interfaces and additional computational capabilities.

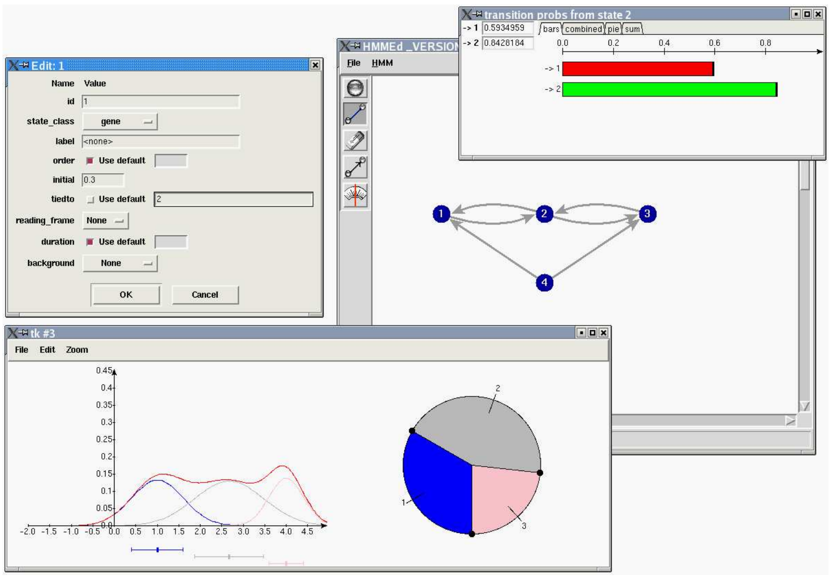
Matlab respectively have become widely popular for ease of use and the little loss of computational effi ciency when one compares a script (a short program of high-level commands) with an all-out Fortran or C implementation for the same computational problem. The GHMM follows this models and thus combines full flexibility (everything is programmable) with effi ciency (everything computationally expensive is coded in C) without locking out potential users, as it is often the case with problem-specifi c GUI applications[3]. Some of the training algorithms are implemented using threads to use micro-parallelism on multi-processor shared memory computers. In applications such as GQL, a Python interface to the standard MPI (Message Passing Interface) library allows the use of distributed computing resources from user code.

4.1 Supporting Teaching

There are two target audiences of learners which we try to address with the GHMM used as a teaching tool. First, students (and scientists) from the application side of things who are interested in building custom applications for their particular statistical modeling or data analysis problem. They are able to use the GHMM from an interactive, high-level language aided by the graphical user interface for editing models. The following real example reads an HMM build for searching for a particular transcription factor binding site, the nucleotide sequence of Human chromosome 16 and through computation of the Viterbi path fi nds putative binding sites.

```
>>> m = HMMOpen("trans-fac-13-hmm.xml")
>>> s = FastaOpen("human-chr16.fa")
>>> v = m.viterbi(s)
>>> print "There are %d transcription factor binding sites" % hits(v)
```

[3]Point in case: the GHMM was originally started because HMM software used for speech recognition implemented all the necessary algorithms, but not in an accessible form; licensing was another issue.



Name	Value
id	1
state_class	gene
label	<none>
order	▪ Use default
initial	0.3
tiedto	Use default 2
reading_frame	None
duration	▪ Use default
background	None

-> 1	0.5934959
-> 2	0.8428184

Figure 10: The HMM editor HMMEd allows graphical design and editing.

It has been our experience that post-Bachelor students were able to implement a simplifi ed variant of our method for analyzing gene expression time-courses, cf. Sec. 2, within one day of a week-long full-day course.

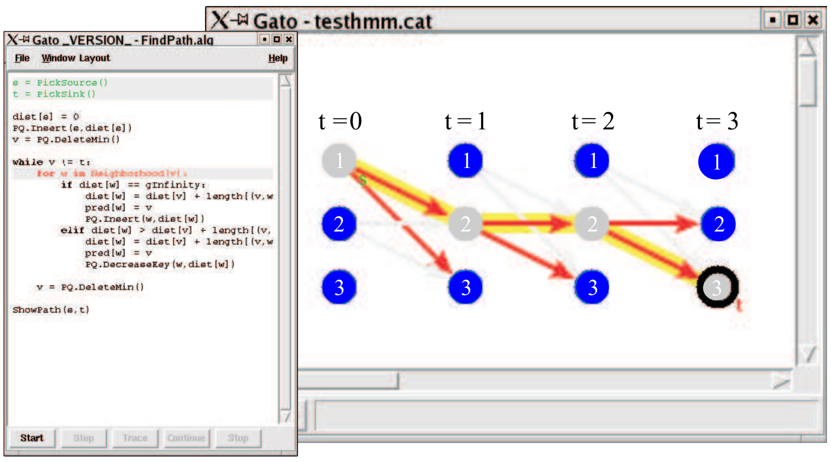
The second group we target are tool developers: students who need to acquire an in-depth understanding of the underlying mathematics to implement variants or extensions of the core algorithms. On one hand they benefi t from the interactive access outlined above during learning, development and testing. On the other hand, we supply semi-automatically generated animations of algorithms which provide visual feedback, see Fig. 11.

5 Summary

The GHMM library provides an essential contribution to scientifi c computing for a widely applicable class of statistical models, namely Hidden Markov Models. Our design and license choices allow effective use from many languages and for different roles of users: the data-analyst, the scripter and the application developer. Driven by the novel approach of modeling biological time-course data with a mixture of HMMs we leveraged our implementational effort into a much more usable end result, with a wider range of applications and a larger, more diverse user base.

Acknowledgments

The GHMM library has been originally conceived at the Zentrum für angewandte Informatik (ZAIK) at the University of Cologne. Thanks to Bernhard Knab, Bernd Wichern, Barthel Steckemetz for the original implementation (not publicly available), Achim Gädke, Peter Pipenbacher and Disa Thorarinsdottir for a lot of help with the creation of the first public release. Many thanks to Ivan G. Costa Filho for maintaining the GQL. Thanks to students from FU Berlin and University of Cologne who used the GHMM in their practical software courses ("Softwarepraktika") in other classes and for their thesis research and all our users for encouragement, bug reports, and code contributions.



```
s = PickSource()
t = PickSink()

dist[s] = 0
PQ.Insert(s,dist[s])
v = PQ.DeleteMin()

while v != t:
    for w in Neighborhood(v):
        if dist[w] == gInfinity:
            dist[w] = dist[v] + length[(v,w
            pred[w] = v
            PQ.Insert(w,dist[w])
        elif dist[w] > dist[v] + length[(v,
            dist[w] = dist[v] + length[(v,w
            pred[w] = v
            PQ.DecreaseKey(w,dist[w])

    v = PQ.DeleteMin()

ShowPath(s,t)
```

Figure 11: Visualization of the Viterbi algorithm for the best alignment of the observation sequence to the HMM. Based on our graph algorithm animation framework Gato [17], we animate the Viterbi algorithm as the shortest path problem in a weighted graph without cycle and non-negative edge weights.

References

[1] Z. Bar-Joseph, G. Gerber, D. K. Gifford, and T. S. Jaakkola. A new approach to analyzing gene expression time series data. *6th Annual Int. Conf. on Research in Comp. Molecular Biology*, 2002.

[2] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, 1998.

[3] R.A. Boyles. On the convergence of the EM algorithm. *JRSS B*, pages 47–50, 1983.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSSB*, 39:1–38, 1977.

[5] M.B. Eisen, P.T. Spellman, P.O Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A.*, 95:14863–8, 1998.

[6] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, and P.O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell.*, 11:4241–57, 2000.

[7] David Kulp, David Haussler, Martin G. Reese, and Frank H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In David J. States, Pamkaj Agarwal, Terry Gaasterland, Lawrence Hunter, and Randall Smith, editors, *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 134–142, Menlo Park, June12–15 1996. AAAI Press. ISBN 1-57735-002-2.

[8] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley, New York, 2000.

[9] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, Inc., New York, Basel, 1988.

[10] C. Nilubol, Q. H. Pham, R. M. Mersereau, and M. J. T. Smith. Translational and rotational invariant hidden markov models for automatic target recognition. In *Proc. Of the SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition VI*, 1998.

[11] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.

[12] M. F. Ramoni, P. Sebastiani, and P. R. Cohen. Bayesian clustering by dynamics. *Mach. Learn.*, 47(1):91–121, April 2002.

[13] M. F. Ramoni, P. Sebastiani, and I. S. Kohane. Cluster analysis of gene expression dynamics. *Proc Natl Acad Sci U S A*, 99(14):9121–9126, Jul 2002.

[14] S. A. Rifkin and J. Kim. Geometry of gene expression dynamics. *Bioinformatics*, 18(9):1176–83, Sep 2002.

[15] A. Schliep, A. Schönhuth, and C. Steinhoff. Using Hidden Markov Models to analyze gene expression time course data. *Bioinformatics*, 19 Suppl 1:I255–I263, Jul 2003.

[16] A. Schliep, C. Steinhoff, and A. Schönhuth. Using Hidden Markov Models to analyze gene expression time course data. *Bioinformatics*, Jul 2004. Proceedings of the 12th International Conference on Intelligent Systems for Molecular Biology. In print.

[17] Alexander Schliep and Winfried Hochstättler. Developing Gato and CATBox with Python: Teaching graph al gorithms through visualization and experimentation. In *Multimedia Tools for Communicating Mathematics*, pages 291–310. Springer-Verlag, Berlin, Heidelberg, 2002.

[18] Philip A. Schrodt. Pattern Recognition of International Crises using Hidden Markov Models. In Diana Richards, editor, *Non-linear Models and Methods in Political Science*. University of Michigan Press, Ann Arbor, MI, 1998.

[19] S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture. *Nat Genet.*, 22:281–5, 1999.

[20] M. L. Whitfield, G. Sherlock, A. J. Saldanha, J. I. Murray, C. A. Ball, K. E. Alexander, J. C. Matese, C. M. Perou, M. M. Hurt, P. O. Brown, and D. Botstein. Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Mol Biol Cell*, 13(6): 1977–2000, Jun 2002.

[21] C.F.J. Wu. On the convergence of the EM algorithm. *Ann. Stat.*, pages 95–103, 1983.

[22] Tetsushi Yada and Makoto Hirosawa. Gene recognition in cyanobacterium genomic sequence data using the hidden Markov model. In David J. States, Pamkaj Agarwal, Terry Gaasterland, Lawrence Hunter, and Randall Smith, editors, *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 252–260, Menlo Park, June12–15 1996. AAAI Press. ISBN 1-57735-002-2.