Gibbs sampling for parsimonious Markov models with latent variables

Ralf Eggeling[1], Pierre-Yves Bourguignon[2,3], André Gohr[1], Ivo Grosse[1]
[1] Martin Luther University Halle-Wittenberg, Germany
[2] Max-Planck-Institute for Mathematics in the Sciences, Germany
[3] Laboratoire de Physique Statistique, UMR 8550 CNRS/ENS/Univ. Paris VI and VII, France

Abstract

Parsimonious Markov models have been recently developed as a generalization of variable order Markov models. Many practical applications involve a setting with latent variables, with a common example being mixture models. Here, we propose a Bayesian model averaging approach for learning mixtures of parsimonious Markov models that is based on Gibbs sampling. The challenging problem is sampling one out of a large number of model structures. We solve it by an efficient dynamic programming algorithm. We apply the resulting Gibbs sampling algorithm to splice site classification, an important problem from computational biology, and find the Bayesian approach to be superior to the non-Bayesian classification.

1 Introduction

Assigning data points to classes based on their similarity to labeled training data is a pervasive task in almost all fields of science. One generally proceeds by assigning a probability to any observation X under the hypothesis that it belongs to some class k. It is customary to use the predictive probabilities $P(X|Y_k)$, which take different forms in the non-Bayesian and Bayesian settings, respectively. Both approaches assume X and training data Y_k to be generated from the same parametric statistical model $\mathcal{M}$. The non-Bayesian approach maps Y_k onto a unique parameter value by means of an estimator (ML, MAP, MP), and recycles this value for computing the predictive distribution

$$P_1(X|Y_k) = P(X|\hat{\theta}_{\mathcal{M}}(Y_k)), \qquad (1)$$

whereas the Bayesian classification relies on the predictive distribution P_2 defined by the following integral over the parameter space:

$$P_2(X|Y_k) = \int P(X|\theta_{\mathcal{M}})P(\theta_{\mathcal{M}}|Y_k)d\theta_{\mathcal{M}}. \qquad (2)$$

Here, the term Bayesian refers to the operation of averaging contributions from the whole model, as opposed to the estimation of a single distribution that is further used for prediction purposes. The probability assignments P_1 and P_2 are not completely unrelated: With $\hat{\theta}_{\mathcal{M}}(Y_k)$ being the mode of one term in the integrand in Equation 2, P_1 can be understood as an approximation of P_2, where only the parameter value contributing most is taken in consideration. P_2, on the other hand, aggregates the contributions of all parameter values according to the support they provide to the training data Y_k.

In many applications, there is also uncertainty about the model $\mathcal{M}$. Viewing $\mathcal{M}$ as a discrete-valued component of the parameter space, P_1 and P_2 are paralleled by P_3 and P_4, respectively, in this setting:

$$P_3(X|Y_k) = P(X|\hat{\theta}_{\hat{\mathcal{M}}(Y_k)}(Y_k)), \qquad (3)$$

which includes a model selection step for computing $\hat{\mathcal{M}}(Y_k)$, and

$$P_4(X|Y_k) = \sum_{\mathcal{M}} P(\mathcal{M}) \cdot \int P(X|\theta_{\mathcal{M}})P(\theta_{\mathcal{M}}|Y_k)d\theta_{\mathcal{M}}, \qquad (4)$$

for the Bayesian classification, which yields a Bayesian model averaging task.

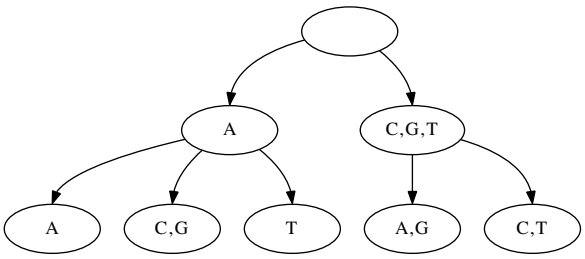


Figure 1: **Example PCT** of depth 2 over DNA alphabet. It encodes the partitioning of all 16 possible sequences into the subsets {AA},{CA,GA},{TA},{AC,AG,AT,GC,GG,GT}, and {CC,CG,CT,TC,TG,TT}.

In both cases, the choice of an appropriate set of candidate models is crucial. Here, we focus on parsimonious Markov models, which have been introduced by Bourguignon (2008) as an extension of variable order Markov models (Rissanen, 1983; Bühlmann and Wyner, 1999). They use parsimonious context trees (PCTs), which differ from traditional context trees in two aspects: (i) a PCT is a balanced tree, i.e. each leaf has the same depth, and (ii) each child node represents an arbitrary subset of the alphabet $\mathcal{A}$, with the additional constraint that sibling nodes form together a partition of $\mathcal{A}$. An example PCT, which forms a partition of strings that cannot be represented by a traditional context tree, is shown in Figure 1.

Many practical applications involve a setting with latent variables or incomplete training data, for example Hidden Markov Models (Rabiner, 1989) or mixture of trees (Meila and Jordan, 2000). In such cases, classification is based on the incomplete versions of P_1 to P_4, i.e. averages thereof against the latent variables. The associated enumeration of all realizations of the latent variables cannot, in general, be performed in an exact yet computationally inexpensive way. As an alternative, the EM algorithm (Dempster et al., 1977) is often used to derive an approximation of $\hat{\theta}_{\mathcal{M}}$. In the Bayesian setting, another option is offered by algorithms that generate samples drawn (at least approximately) from the posterior distribution of the parameter, $P(\theta_{\mathcal{M}}|Y_k)$. Predictive probabilities such as P_2 and P_4 can then be derived by substituting a discrete approximation of the integral using the generated parameter values.

One of the simplest models that uses latent variables is a C-component mixture model. A recent algorithm for learning mixtures of inhomogeneous parsimonious Markov models is based on the MAP principle and thus uses a modified EM algorithm (Gohr et al., 2012). Here, we derive a Gibbs sampler for sampling from the posterior distribution of mixtures of inhomogeneous parsimonious Markov models. We study the convergence behavior of the algorithm and evaluate its classification performance compared to the corresponding EM algorithm.

2 Model and prior

The data are symbolic sequences of fixed length L over the alphabet $\mathcal{A}$. We denote a single symbol by X, a sequence of length L by $\vec{X} = (X_1, \ldots, X_L)$ and a data set of N sequences by $\mathbf{X} = (\vec{X}_1, \ldots, \vec{X}_N)$.

2.1 Parsimonious Markov model

Here, we focus on inhomogeneous parsimonious Markov models (parsMMs) of order D, which use potentially different *parsimonious context trees* (PCTs) for each position. A PCT is a rooted, balanced tree, which we subsequently denote by τ. $\mathcal{T}_D^{\mathcal{A}}$ is the set of all possible trees for a given alphabet $\mathcal{A}$ and depth D. Each node of a PCT is labeled with a non-empty subset of $\mathcal{A}$, except for the root, which is labeled by the empty subset. The set of labels of all children of an arbitrary inner node forms a partition of $\mathcal{A}$. Starting from a leaf, building the cartesian product of all subsets found on the path to the root defines a set of sequences, which is the *context* encoded by that leaf.

The example PCT in Figure 1 encodes the contexts $\{A\} \times \{A\}, \{C, G\} \times \{A\}$, $\{T\} \times \{A\}$, $\{A, G\} \times \{C, G, T\}$, and $\{C, T\} \times \{C, G, T\}$.

A parsimonious Markov model of order D for a sequence $\vec{X}$ of length L involves exactly L PCTs, denoted by $\vec{\tau} = (\tau_1, \ldots, \tau_L)$. For the ease of presentation, we exclude from the following discussion the first D PCTs, which have an increasing maximal depth of $0, \ldots, D-1$. We

denote a single context as $\mathbf{w}$, and all contexts represented by a specific parsimonious context tree τ by $\mathcal{C}_\tau$. For a given PCT τ, we denote the conditional probability of observing a symbol $a \in \mathcal{A}$, given that the concatenation of the preceding D symbols is in $\mathbf{w}$, as $\theta^\tau_{\mathbf{w}a}$. We denote all parameters of a single position in the parsMM by $\Theta = \left(\tau, (\vec{\theta}^{\tau'})_{\tau' \in \mathcal{T}_D^{\mathcal{A}}}\right)$, using the product space formulation of Carlin and Chib (1995) in order to ensure a fixed dimensionality of the parameter space. We further combine the parameters of all positions in the parsMM by $\vec{\Theta} = (\Theta_1, \ldots, \Theta_L)$. The likelihood function of a parsMM is given as

$$P(\mathbf{X}|\vec{\Theta}) = \prod_{\ell=1}^{L} \prod_{\mathbf{w} \in \mathcal{C}_{\tau_\ell}} \prod_{a \in \mathcal{A}} \left(\theta^{\tau_\ell}_{\ell \mathbf{w} a}\right)^{N_{\ell \mathbf{w} a}}, \qquad (5)$$

where $N_{\ell \mathbf{w} a}$ is the number of occurrences of symbol a at position ℓ in all sequences in data set $\mathbf{X}$ where the symbols from position $\ell - D$ to $\ell - 1$ are an element of $\mathbf{w}$. We define a prior for the parsimonious Markov model by

$$P(\vec{\Theta}) = P(\vec{\tau}) \prod_{\ell=1}^{\ell} \prod_{\tau' \in \mathcal{T}^{\mathcal{A}}_{\ell,D}} \prod_{\mathbf{w} \in \mathcal{C}_{\tau'}} P(\vec{\theta}^{\tau'}_{\ell \mathbf{w}}) \qquad (6)$$

where $P(\vec{\theta}^{\tau'}_{\ell \mathbf{w}})$ is a Dirichlet distribution with hyperparameters $\vec{\alpha}^{\tau'}_{\ell \mathbf{w}}$. For the case studies in this work, we further restrict the parameter prior to a symmetric Dirichlet distribution. Following the equivalent sample size (ESS) concept (Heckerman et al., 1995), we obtain a natural computation of the pseudocounts from the ESS that is inspired by Bayesian networks, namely $\alpha^\tau_{\ell \mathbf{w} a} = \frac{\mathrm{ESS}|\mathbf{w}|}{|\mathcal{A}|^{D+1}}$. We specify the structure prior over all L PCTs in the model by

$$P(\vec{\tau}) \propto \prod_{\ell=1}^{L} \kappa^{|\mathcal{C}_{\tau_\ell}|}. \qquad (7)$$

It depends on one hyperparameter $\kappa \in (0, \infty)$, which can be used to influence the number of leaves and thus the complexity of the model, interpolating between the two special cases: When $\kappa \to +\infty$, the maximal tree, which represents a full order Markov model, receives a prior probability of one. Conversely, when if $\kappa \to 0$, only the minimal tree, which represents an independence model, receives prior support.

2.2 Mixture model

We consider a C-component mixture model as a typical instance of a model with latent variables. The assignment of each of the N sequences to one of the C components is specified by the latent vector $\vec{u} = (u_1, \ldots, u_N) \in \{1, \ldots, C\}^N$. Each component of the mixture model is an inhomogeneous parsimonious Markov model, so the complete set of parameters is denoted by $\mathbf{\Theta} = (\vec{\pi}, \vec{\Theta}_1, \ldots, \vec{\Theta}_C)$, where $\vec{\pi} = (\pi_1, \ldots, \pi_C)$, and π_c contains the probability of the c-th mixture component. The likelihood of the mixture model is thus given as

$$P(\mathbf{X}, \vec{u}|\mathbf{\Theta}) = P(\mathbf{X}|\vec{u}, \mathbf{\Theta}) P(\vec{u}|\mathbf{\Theta}), \qquad (8)$$

where

$$P(\vec{u}|\mathbf{\Theta}) = \prod_{i=1}^{N} \pi_{u_i}, \qquad (9)$$

and

$$P(\mathbf{X}|\vec{u}, \mathbf{\Theta}) = \prod_{c=1}^{C} P(\mathbf{X}_{\{\vec{u}=c\}}|\vec{\Theta}_c) \qquad (10)$$

with

$$\mathbf{X}_{\{\vec{u}=c\}} = (X_i)_{i|u_i=c} \qquad (11)$$

denoting all sequences that are assigned to component c by the latent variables $\vec{u}$. The prior of the full mixture model factorizes as

$$P(\mathbf{\Theta}) = \prod_{c=1}^{C} P(\vec{\Theta}_c). \qquad (12)$$

For the sake of simplicity, we set $\vec{\pi}$ in this work externally to a uniform distribution.

3 Gibbs sampling

We now shall generate a sample from the distribution $P(\mathbf{\Theta}, \vec{u}|\mathbf{X})$, from which $P(\mathbf{\Theta}|\mathbf{X})$ for use in Equation 4 can be further derived by marginalization.

Sampling from $P(\mathbf{\Theta}, \vec{u}|\mathbf{X})$ directly is intractable, but approximate sampling techniques

$$\forall_{c=1}^{C}\forall_{\ell=1}^{L}: \text{ sample } \quad \tau_{c\ell}^{(t)} \quad \text{ from } \quad P(\tau_{c\ell}|\vec{u}^{(t-1)},\mathbf{X}) \tag{13}$$

$$\forall_{i=1}^{C}\forall_{\ell=1}^{L}\forall_{\mathbf{w}\in\mathcal{C}_{\tau_{c\ell}}}: \text{ sample } \quad \theta_{c\ell w}^{\tau_{c\ell}\ (t)} \quad \text{ from } \quad P(\theta_{c\ell w}^{\tau_{c\ell}}|\tau_{c\ell}^{(t)},\vec{u}^{(t-1)},\mathbf{X}) \tag{14}$$

$$\forall_{i=1}^{N}: \text{ sample } \quad u_i^{(t)} \quad \text{ from } \quad P(u_i|\mathbf{\Theta}^{(t)},\vec{X}_i) \tag{15}$$

Figure 2: **Sampling steps** for the t-th iteration of the Gibbs sampler.

are available. We focus here on Gibbs sampling (Geman and Geman, 1984; Casella and George, 1992), where each parameter is iteratively updated using its conditional distribution given the current value of the other parameters. Samples form a realization of a Markov chain, whose stationary distribution is the posterior.

We sample in the t-th iteration $\mathbf{\Theta}^{(t)}$ from $P(\mathbf{\Theta}|\vec{u}^{(t-1)},\mathbf{X})$ and $\vec{u}^{(t)}$ from $P(\vec{u}|\mathbf{\Theta}^{(t)},\mathbf{X})$. The conditional probability of the parameters given the latent variables decomposes to

$$P(\mathbf{\Theta}|\vec{u}^{(t-1)},\mathbf{X}) = \prod_{c=1}^{C}\prod_{\ell=1}^{L} P(\tau_{c\ell},\vec{\theta}_{c\ell}|\vec{u}^{(t-1)},\mathbf{X}). \tag{16}$$

For each compontent c and each position ℓ (and thus omitting here both indices for the sake of convenience), we use the idea of variable grouping (Liu, 2001) and sample $(\tau,\vec{\theta}^{\tau})^{(t)}$ jointly from $P(\tau,\vec{\theta}^{\tau}|\vec{u}^{(t-1)},\mathbf{X})$ instead of sampling from each single conditional distribution separately. This is achieved in a hierarchical manner, by decomposing the joint conditional probability distribution into $P(\tau|\vec{u}^{(t-1)},\mathbf{X})P(\vec{\theta}^{\tau}|\tau,\vec{u}^{(t-1)},\mathbf{X})$. We first sample $\tau^{(t)}$ w.r.t. its conditional distribution given $(\vec{u}^{(t-1)},\mathbf{X})$, and then $\vec{\theta}^{\tau\ (t)}$ from its full conditional distribution, $P(\vec{\theta}^{\tau}|\tau^{(t)},\vec{u}^{(t-1)},\mathbf{X})$, under which the components of $\vec{\theta}^{\tau}$ are independent. Similarly, the components of $\vec{u}$ are conditionally independent given the other parameters and the data.

As a result, we obtain the sampling scheme that is shown in Figure 2. The remaining task is to efficiently sample from the specified conditional distributions, which we discuss in the following sections. Whereas sampling of latent variables and probability parameters are comparatively simple, the particular challenge lies in the sampling of the PCTs.

3.1 Structure sampling

In this section, we focus on the sampling of a PCT structure in order to perform the sampling step 13. The probability of a particular tree structure given data (step 13) is

$$P(\tau_{c\ell}|\vec{u},\mathbf{X}) \propto \prod_{\mathbf{w}\in\mathcal{C}_{\tau_{c\ell}}} \kappa\frac{\mathcal{B}(\vec{N}_{c\ell\mathbf{w}}+\vec{\alpha}_{c\ell\mathbf{w}})}{\mathcal{B}(\vec{\alpha}_{c\ell\mathbf{w}})}, \tag{17}$$

where $\mathcal{B}$ denotes the multinomial beta-function. Hence, the probability decomposes into a product of scores for each context, i.e. leaf scores for each leaf in the PCT. Each leaf score is itself a marginal likelihood for the particular context multiplied with the structure prior hyperparameter κ. While the probability for a given tree can be computed easily, the challenge lies in sampling one out of a super-exponential number (with respect to model order and alphabet size) of possible PCTs, without computing the probability for every single tree explicitely.

To this extent, we propose a dynamic programming algorithm, which is inspired by that of Bourguignon (2008) and Volf and Willems (1994) for finding the PCT or CT that maximizes a given score.

The algorithm runs on a specific data structure, the extended tree, of which one subtree is shown in Figure 3. In contrast to a PCT, the children of a node of an extended tree do not form a partition of $\mathcal{A}$, but rather encompass all elements of $\mathcal{P}(\mathcal{A})\setminus\{\emptyset\}$.

We construct top-down an extended tree of depth D, and then sample in a bottom-up traversal a regular PCT by taking – for each node n – one out of two possible actions:

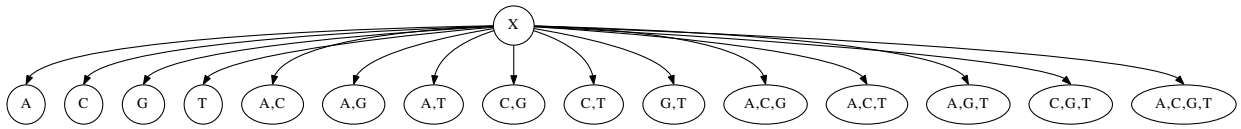


Figure 3: **Extended tree**. The picture depicts the structure of an arbitrary inner node (labeled with X) and its children in the extended tree over a four letter alphabet. The labels of all children of a node form the power set of $\mathcal{A}$, with the exception of the empty set.

(i) If n is a leaf (representing context $\mathbf{w}$), we compute the score $\kappa \frac{\mathcal{B}(\vec{N}_{c\ell\mathbf{w}}+\vec{\alpha}_{c\ell\mathbf{w}})}{\mathcal{B}(\vec{\alpha}_{c\ell\mathbf{w}})}$, and assign it to n.

(ii) If n is an inner node, we first compute the probability of each *valid choice* of children of n, where a valid choice is a set of children, whose labels form a partition of $\mathcal{A}$ and the score of a valid choice is simply the product of the scores of the children contained in it. Next, one valid choice is sampled according to the computed probability distribution. The probability of this sampled set of children becomes the score of n. The remaining children of n, which do not belong to the sampled set, and all subtrees below are discarded. Hence n becomes the root of a subtree that satisfies the characteristics of a PCT and has a score assigned to it.

We obtain a complete PCT, once we have sampled a valid choice of children of the root of the extended tree.

The algorithm samples correctly from the posterior distribution for the following reasons: First, when sampling the children of a particular node, the scores of all potential children are already available. In step (ii), we can safely assume that for each inner node n, each child (Figure 3) is either a leaf, in which case we have obtained its score by step (i), or the root of a subtree that already satisfies the characteristics of a PCT and has a score assigned to it. Second, the subtree rooted at an arbitrary node n and subtrees rooted at the siblings of n are conditionally independent given the labels on the path from n to the global root of the PCT. This information is available at any time due to the top-down construction of the extended tree.

The time complexity of the algorithm is given by the number of valid choices multiplied by the number of inner nodes in the extended tree, more precisely $\mathcal{O}\left(B_{|\mathcal{A}|}\left(2^{|\mathcal{A}|}-1\right)^{D-1}\right)$, where $B_{|\mathcal{A}|}$ is the Bell number (Rota, 1964).

3.2 Parameters

The probability of the conditional probability parameters of a given context of a given PCT structure (step 14) is a Dirichlet distribution with the hyperparameters being a sum of counts and pseudocounts, that is,

$$P(\theta_{c\ell w}^{\tau_{c\ell}}|\tau_{c\ell},\vec{u},\mathbf{X}) = \mathrm{Dir}(\theta_{c\ell w}^{\tau_{c\ell}}|\vec{N}_{c\ell\mathbf{w}}+\vec{\alpha}_{c\ell\mathbf{w}}). \quad (18)$$

3.3 Latent variables

The conditional probability distribution of the latent variables (step 15) is merely a fraction of likelihood values:

$$P(u_i|\mathbf{\Theta},\vec{X}_i) = \frac{\pi_{u_i} P(\vec{X}_i|\vec{\Theta}_{u_i})}{\sum_{c=1}^{C} \pi_c P(\vec{X}_i|\vec{\Theta}_c)} \quad (19)$$

It is noteworthy that given the parameters and the data, the components of $\vec{u}$ are mutually independent. In addition, the conditional distribution of each component u_i depends on the data only through the sequence X_i.

4 Case studies

In order to evaluate the method on real world data, we apply it to the problem of splice site classification, which is an important task in computational biology. Splice sites are short DNA sequences that contain a `GT` dinucleotide. However, not all sequences containing a `GT` dinucleotide are functional splice sites. The statistical problem is to distinguish functional splice sites from non splice sites by modeling the variable nucleotides left and right of the `GT`.

Table 1: **Data sets** used for the classification experiment, consisting of experimentally verified positive and negative data.

	training	test
splice donor sites	`pos_train`	`pos_test`
non splice sites	`neg_train`	`neg_test`

We use the splice donor site data sets of Yeo and Burge (2004), which consist of experimentally verified splice donor sites and non splice sites (Table 1). The conserved `GT` dinucleotide has been removed already, the remaining sequences of length of $L = 7$ over $\mathcal{A} \in \{A, C, G, T\}$ are diverse, which makes the classification problem challenging (Yeo and Burge, 2004). In the following, we use a mixture of $C = 2$ parsMMs of depth $D = 2$ with an ESS $= 16$ for each component for learning from the functional splice sites.

4.1 Convergence

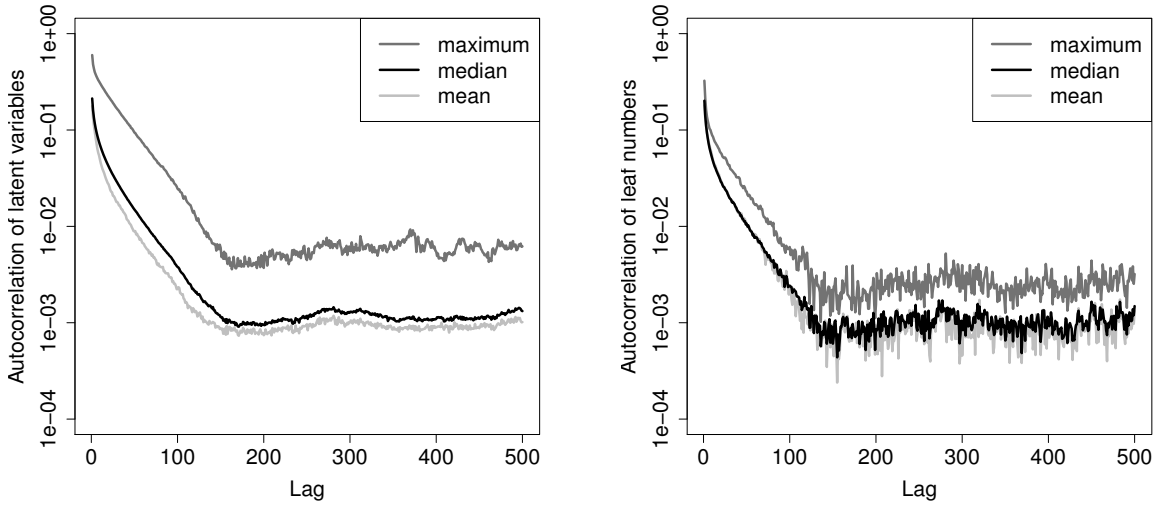
Gibbs sampling, like any MCMC method, samples from the target distribution only asymptotically. It is therefore important to study the convergence rate of the algorithm in order to control the quality of the approximation brought by the sample. In practice, a number of first samples is generated but discarded, thereby skipping the burn-in phase of the sampler. We investigate the convergence for a data set of size $N = 500$, since we use data sets of the same size for a classification study in section 4.2. After initializing each latent variable by sampling from the uniform distribution $(0.5, 0.5)$, we perform 10^4 iterations of the Gibbs sampler. In each iteration step, we store the sampled value of each of the 500 latent variables. In absence of a simple notion of correlation among PCTs, we focus here on the number of leaves of the PCTs. The tree of the first position of both components can be neglected, since it always consists of one leaf. So we store only the number of leaves of each PCT at position 2-7 in both components, yielding 12 additional variables per iteration, which makes 512 variables in total. Since the space of actively used probability parameters changes from iteration step to iteration step, we do not measure their convergence behavior.

Next, we compute the autocorrelation function for each of the 512 variables with lags of 1 to 500 from the 10^4 iterations. We repeat the process 10^2 times with different initializations and average the autocorrelation coefficients for each variable. We project the results to six curves in the following way. First, we investige the latent variables and the PCT complexity separately. Second, we plot for each lag the (i) maximum, (ii) mean, and (iii) median of the absolute autocorrelation, which is shown in Figure 4. In both cases, we observe an approximately exponential decay of the autocorrelation up to a lag of approximately 150. For larger lags, it differs only slighty, even though there is a small increase between lag 250 and 350. We finally conclude that 200 iteration steps is the minimal length of the burn-in phase, since all samples before that are still correlated to the random initialization. For the following classification studies, we use a burn-in phase of 1000 iteration steps.

4.2 Classification

After having evaluated convergence behavior of the Gibbs sampling algorithm, we study how well it performs in practice compared to an EM algorithm when facing the task of classification. We intent to classify splice donor sites (positive) against non splice sites (negatives). To this extend, we use an independence model for the negative class. In the following study we use all four data sets that are depicted in Table 1. All data sets except `pos_train` are kept fixed during the entire experiment. `pos_train`, which we utilize to train the mixture model, is a random sample of 500 sequences from the original training data set of Yeo and Burge (2004).

Using these four data sets, we train a classifier by (a) training an independence model on `neg_train` and (b) training a two-component mixture of parsimonious Markov models on `pos_train` by (i) Gibbs sampling and (ii) the EM algorithm. In case of (i), we obtain a series of 10^3 parameter sets that are samples from the posterior distribution after the burn-in phase,



(a) Hidden variables

(b) Number of Leaves

Figure 4: **Autocorrelation coefficients** of latent variables and number of leaves of the parsimonious context trees in logarithmic scale. Figure 4a depicts for each lag the mean, median and maximum of the autocorrelation over the 500 latent variables. Figure 4a depicts the same statistics of the autocorrelation of the PCT leaf number over the 12 nontrivial PCTs in the mixture model. In all cases, the autocorrelation shows a nearly exponential decay with a decay constant of approximately 1.7 until it remains stable at low values at lag 150-200.

where in only every hundreth iteration a parameter set is stored. We start the Gibbs sampler ten times and use all the 10^4 parameter sets for a classification according to Equation 4. In case of (ii), we run the EM algorithm of Gohr et al. (2012) until the difference in the posterior is smaller than 10^{-6}. We repeat the procedure ten times with different initializations, and use the parameter set from the run that yields the highest posterior in order to classify according to Equation 3. In both cases, we classify all sequences in `pos_test` and `neg_test` and compute the area under the ROC curve (AUC). We repeat the entire procedure with 20 different samples of size 500 from the training data set, and compute the mean AUC and its standard error.

The classification performance might be heavily influenced by the model complexity. So we repeat the study with different values of the structure prior constant κ. By varying κ from 10^{-50} to 10^{10}, we cover the whole scale of complexity that a parsMM(2) can represent, from independence model, obtained when all PCTs contain only one leaf, to a second order inhomogeneous Markov model, obtained when all PCTs have the maximal number of leaves.

The results are shown in Figure 5. As we expected, the classification performance depends heavily on the complexity. However, for all levels of complexity, the Bayesian classification based on Gibbs sampling outperforms the non-Bayesian classification that uses the EM algorithm by a wide margin. Moreover, the standard errors are substantially smaller, indicating that the Bayesian classification also yields more stable results.

4.3 Conclusions

We dervived a Gibbs sampling algorithm that samples PCT structures and corresponding probability parameters from the posterior distribution of parsimonious Markov models in a setting with latent variables. We studied the convergence of the algorithm and found 1000

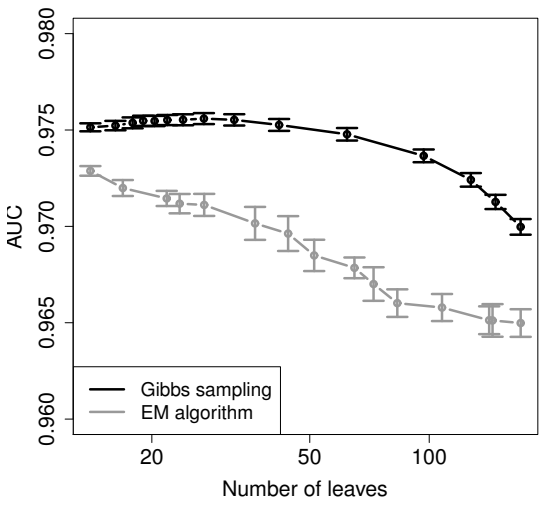


Figure 5: **Mean AUC** of a repeated holdout classification experiment for Gibbs sampler and EM algorithm for different model complexities. Each error bar shows the double standard error. Gibbs sampler outperforms EM algorithm for all possible model complexities.

sampling steps to be a safe estimation of the length of the burn-in phase. We applied the algorithm to splice site classification and observed the Gibbs sampler outperforming the EM algorithm for every model complexity. These results suggest that the Bayesian learning approach might also be useful when parsimonious Markov models are combined with other models that also involve latent variables.

Acknowledgments

This work was funded by *Reisestipendium des allg. Stiftungsfonds der MLU Halle-Wittenberg*, *MPG/CNRS SysBio* research program, and DFG (grant no. GR-3526_1-1). We thank Petri Myllymäki, Teemu Roos, and Antti Honkela for valuable discussions.

References

Pierre-Yves Bourguignon. 2008. *Parcimonie dans les modèles markoviens et applications à l'analyse des séquences biologiques*. Ph.D. thesis, Université Evry Val d'Essonne.

P. Bühlmann and A.J. Wyner. 1999. Variable length Markov chains. *Annals of Statistics*, 27:480–513.

B.P. Carlin and S. Chib. 1995. Bayesian Model Choice via Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(3):473–484.

G. Casella and E.I. George. 1992. Explaining the Gibbs Sampler. *The American Statistician*, 46:167–174.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

S. Geman and D. Geman. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 6, pages 721–741.

A. Gohr, S. Posch, and I. Grosse. 2012. Mixtures of Parsimonious Markov Models. Technical Report 2012/2, Institute of Computer Science, Martin Luther University Halle-Wittenberg, Germany.

G. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20:197–243.

Jun S. Liu. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics.

M. Meila and M.I. Jordan. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48.

L.R. Rabiner. 1989. A tutorial on Hidden Markov-Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–268.

Jorma Rissanen. 1983. A universal data compression system. *IEEE Trans. Inform. Theory*, 29(5):656–664.

G.C. Rota. 1964. The Number of Partitions of a Set. *Amer. Math. Monthly*, 71:498–504.

P. Volf and F. Willems. 1994. Context maximizing: Finding MDL decision trees. In *15th Symp. Inform. Theory Benelux*, pages 192–200, May.

G. Yeo and C.B. Burge. 2004. Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals. *Journal of Computational Biology*, 11(2/3):377–394.