Towards a Novel Probabilistic Graphical Model of Sequential Data: A Solution to the Problem of Structure Learning and an Empirical Evaluation

Marco Bongini and Edmondo Trentin

Dipartimento di Ingegneria dell'Informazione,
Università degli Studi di Siena, Siena, Italy
{bongini,trentin}@dii.unisi.it

Abstract. This paper develops a maximum pseudo-likelihood algorithm for learning the structure of the dynamic extension of Hybrid Random Field introduced in the companion paper [5]. The technique turns out to be a viable method for capturing the statistical (in)dependencies among the random variables within a sequence of patterns. Complexity issues are tackled by means of adequate strategies from classic literature on probabilistic graphical models. A preliminary empirical evaluation is presented eventually.

Keywords: Probabilistic graphical model, Hidden Markov model, Hybrid Random Field, Sequence Classification.

1 Introduction

The notion of Dynamic Hybrid Random Field (DHRF) has been introduced in the companion paper [5], relying on the strict-sense definition of HRF given in [2] and summarized in [5]. The reader is referred to these bibliographical sources for all major concepts and the notation of standard HRFs and DHRFs. In short, an HRF is thence a model of the joint probability of a set of random variables that can be expressed in terms of a collection of Bayesian networks (BN) which possess the "modularity property".

This paper develops a maximum pseudo-likelihood algorithm for learning the structure of a Dynamic Hybrid Random Field from data. The technique is devised in Section 2. Once learning of the structure is completed, the algorithm for parameter learning presented in the companion paper [5] (coherently aimed at the maximization of the same criterion) may be applied.

The remains of the paper go as follows. Section 3 reports on a preliminary empirical evaluation of DHRFs involving synthetic sequences of patterns drawn from specific probability distributions. Therein, DHRFs are compared with Dynamic Bayesian Networks (DBN). Section 4 draws some preliminary conclusive remarks.

N. Mana, F. Schwenker, and E. Trentin (Eds.): ANNPR 2012, LNAI 7477, pp. 82–92, 2012.

2 Structure Learning

Structure learning in q-th HRFs within the DHRF is the problem of learning, for each variable X_i, what other variables appear as nodes in the Bayesian network $BN_{q,i}$, and what edges are contained in the directed acyclic graph (DAG) $\mathcal{G}_{q,i}$. In other words, this means learning the structure of each Markov blanket $\mathcal{MB}_{q,i}(X_i)$ within q-th hybrid random field. While parameter learning assumes that the Markov blanket of each variable has previously been fixed, the aim of structure learning is to identify each Markov blanket and to determine its graphical structure.

We now present a heuristic structure learning algorithm for DHRFs, which we will call *dynamic Markov blanket merging* (DMBM). The aim of DMBM is to find an assignment of Markov blankets $\mathcal{MB}_{q,1}(X_1), \ldots, \mathcal{MB}_{q,n}(X_n)$ to the nodes $X_1, \ldots, X_n$ (within q-th HRF) that maximizes the model pseudo-likelihood given a dataset. The basic idea behind DMBM is to start from a certain assignment of neighbors to the variables of the model, learn the local Bayesian networks of the model, and then to iteratively refine the assignment so as to come up with Markov blankets that increase the model pseudo-likelihood with respect to the previous assignment. This iterative procedure stops when no further refinement of the Markov blankets assignment increases the value of the pseudo-likelihood. In other words, DMBM is nothing but a local search algorithm exploring a space of possible Markov blanket assignments to the observable variables for each state of the DHRF. An important problem to consider in this connection is the dimensionality of the search space. If we allow the search space to contain all possible Markov blanket assignments, the size of the space will be intractable: if n is the number of observable variables, for each variable there are 2^{n-1} possible Markov blankets, which means that the size of the search space will be $Q \cdot n \cdot 2^{n-1}$. Clearly, it is not possible to explore such a state space exhaustively. For this reason, DMBM is designed to explore only a small region of that space, as we are going to see.

In order to develop the algorithm, we specify three components: a way to produce the initial assignment, i.e. a model initialization strategy; a way to refine a given assignment so as to produce an alternative assignment, i.e. a search operator; a way to evaluate a given assignment, i.e. an evaluation function. Also, a general technique for learning the structure of Bayesian networks is required throughout the iterations of DMBM. The structure learning algorithm for BNs used in the paper, relying on the minimum description length principle, may be found in [2]. The next sections outline DMBM in detail.

2.1 Initial Segmentation

The first step of the algorithm relies on an initial segmentation of input sequences and on the creation of initial training sets for all HRFs $\mathcal{H}_1, \ldots, \mathcal{H}_Q$. These initial datasets are then used (see the next sections) for a heuristic initialization and refinement of the structures of the HRFs. Once the structures are learnt, the whole DHRF is used in order to obtain a new, improved segmentation of the

original training sequences and the algorithm is iterated in an EM-fashion (e.g., relying on the Viterbi algorithm), yielding increasingly finer segmentations and structures. The initial segmentation can be obtained in several ways, e.g. relying on traditional machine learning approaches to sequence modeling. The simplest approach we consider herein involves a "linear" segmentation. Let us assume that a training sequence $O = O_1, \ldots, O_T$ is given, which has been generated by a sequence of latent variables $q_1, \ldots, q_L$ (note that $L << T$, in general). The linear segmentation splits O into L ordered subsequences having equal length $\ell = \lfloor T/L \rfloor$ (where $\lfloor . \rfloor$ denotes the usual floor function). State-specific datasets $\mathbf{D}_1, \ldots, \mathbf{D}_Q$ (initially empty) are built up from all the training sequences by merging them with the input observations that belong to the corresponding subsequences, i.e. $\mathbf{D}_1 = \mathbf{D}_1 \cup \{O_1, \ldots, O_\ell\}$, $\mathbf{D}_2 = \mathbf{D}_2 \cup \{O_{\ell+1}, \ldots, O_{2\ell}\}$, $\ldots$, $\mathbf{D}_Q = \mathbf{D}_Q \cup \{O_{(L-1)\ell+1}, \ldots, O_T\}$.

2.2 HRFs Initialization

The way DMBM produces an initial assignment (i.e., structure) for each $\mathcal{H}_q$ is by choosing an initial size k of the neighborhoods, and then by assigning as neighbors in $\mathcal{H}_q$ to each variable X_i those k variables that achieve the highest scores on the χ^2 dependence test over $\mathbf{D}_q$ with respect to X_i. The intuitive motivation for this choice is that a neighborhood containing variables that are more strongly correlated to X_i is more likely to capture the Markov blanket of X_i in $\mathcal{H}_q$ than a neighborhood containing variables that are only weakly correlated to X_i. Given the neighborhoods in $\mathcal{H}_q$, an assignment of Markov blankets to the respective variables is obtained by learning (both the structure and the parameters of) a Bayesian network $BN_{q,i}$ for each variable X_i, where $BN_{q,i}$ contains X_i together with its neighborhood $\mathcal{N}_q(X_i)$. As we said, in order to learn the structure of the local Bayesian networks, we use the technique described in [2].

2.3 Search Operator

Given current assignments of Markov blankets to the model variables in $\mathcal{H}_1 \ldots, \mathcal{H}_Q$, where the assignments are given by the Markov blankets $\mathcal{MB}_{q,1}(X_1), \ldots, \mathcal{MB}_{q,n}(X_n)$ specified by the networks $BN_{q,1}, \ldots, BN_{q,n}$ for $q = 1, \ldots, Q$, new assignments are obtained as follows. For each HRF $\mathcal{H}_q$ and for each variable X_i, we construct the set $\mathcal{U}_{q,i}$ as the union of $\mathcal{MB}_{q,i}$ with the Markov blankets of X_i in all graphs $\mathcal{G}_{q,j}$ such that X_i appears in $\mathcal{G}_{q,j}$ within the current assignment. Given the sets $\mathcal{U}_{q,1}, \ldots, \mathcal{U}_{q,n}$, we first check whether the cardinality of each $\mathcal{U}_{q,i}$ does not exceed a certain threshold k^*, and then we construct a new set of Bayesian networks $BN'_{q,1}, \ldots, BN'_{q,n}$ such that, for each q and for each i, if $|\mathcal{U}_{q,i}| \leq k^*$, then $BN'_{q,i}$ is the network learned by using the set $\mathcal{N}_q(X_i) = \mathcal{U}_{q,i}$ as neighborhood of X_i in $\mathcal{H}_q$, whereas, if $|\mathcal{U}_{q,i}| > k^*$, then $BN'_{q,i} = BN_{q,i}$. Given the networks $BN'_{q,1}, \ldots, BN'_{q,n}$, a new assignment of Markov blankets to the variables $X_1, \ldots, X_n$ in $\mathcal{H}_q$ is obtained in the following manner. For

each X_i and for each q, we compare the value $\sum_{j=1}^{|\mathbf{D}_q|} \log P(x_{i_j}|mb_{q,i_j}(X_i))$ to the value $\sum_{j=1}^{|\mathbf{D}_q|} \log P(x_{i_j}|mb'_{q,i_j}(X_i))$. This values are, respectively, the conditional log-likelihoods of X_i given its Markov blanket, as determined on the one hand by Bayesian network BN_i, and on the other hand by Bayesian network $BN'_{q,i}$. If the latter value is higher than the former, that is if the conditional log-likelihood of X_i given its Markov blanket is increased by replacing $\mathcal{MB}_{q,i}(X_i)$ with $\mathcal{MB}'_{q,i}(X_i)$, then $\mathcal{MB}'_{q,i}(X_i)$ will be the Markov blanket of X_i in $\mathcal{H}_q$ in the new assignment, otherwise X_i will be assigned again $\mathcal{MB}_{q,i}(X_i)$.

2.4 Evaluation Function

An assignment of Markov blankets to the variables is evaluated by measuring the DHRF pseudo-log-likelihood, that is the logarithm of $P^*(O|\mathcal{H})$. Locally (at a specific HRF $\mathcal{H}_q$ level) maximization of the latter is guaranteed by maximization of an evaluation function in the form of a pseudo-log-likelihood of q-th model given the corresponding dataset $\mathbf{D}_q$, namely:

$$\log P^*(\mathbf{D}_q|\mathcal{H}_q) = \sum_{j=1}^{|\mathbf{D}_q|} \sum_{i=1}^{n} \log P(X_i = x_{i_j}|mb_{q,i_j}(X_i)) \tag{1}$$

Actually, by building an alternative assignment from the current one we implicitly evaluate the new assignment. In fact, the new assignment will differ from the old one only if there is at least one variable X_i such that the new Markov blanket of X_i in $\mathcal{H}_q$ increases the variable conditional log-likelihood given the Markov blanket. Given the definition of the pseudo-log-likelihood function and the way our search operator works, an increase in any one of the n local log-likelihoods of any HRFs (within the DHRF) ensures an increase in the global pseudo-log-likelihood of the DHRF. The reason is clearly that the search operator works in a modular fashion, that is the way each Markov blanket $\mathcal{MB}_{q,i}(X_i)$ is modified by the operator is such that the change does not affect any other Markov blanket in the model. Therefore, after we build a new assignment, it will be sufficient to compare it to the old one in order to know whether it increases the model pseudo-likelihood: if the two assignments are different, then we can endorse the new one as being better, otherwise we keep the old one and stop the search.

2.5 Iteration

Once the structures have been learned, parameters of the DHRF can be estimated as explained in the companion paper [5]. The DHRF trained this way can be used to perform a new, more precise segmentation of the training sequences, building new state-specific training sets $\mathbf{D}_1, \ldots, \mathbf{D}_Q$ from the corresponding subsequences, and the whole structure learning procedure can be carried out all over again in order to obtain improved structures for $\mathcal{H}_1, \ldots, \mathcal{H}_Q$. This iterative scheme completes the dynamic Markov blanket merging algorithm.

3 Empirical Evaluation

In order to evaluate the model empirically, we realized a simulator in JAVA, developed over the *JProGraM* library for HRFs [1]. We compared the DHRF with its natural competitor, the Dynamic Bayesian Network (DBN) (reviewed in short in [5]). The evaluation concerns only synthetic data. In section 3.1 we describe the data generation process, and in section 3.2 we present the results of the experiments.

3.1 Synthetic Data Generation

The data generation process is in two steps: (i) creation of a model which specifies the set of probability laws underlying the data distribution; (ii) generation of the data, drawn from these laws. In practice, we proceed with the generation of a random DHRF, selecting for each random variable in each HRF a random Markov Blanket, and then selecting a random structure for the relative BN. Then, we generate the corresponding, random conditional probability tables (CPTs). Afterwards, we use the resulting DHRF as a probabilistic generative model, drawing the random sequences which constitute our dataset from the joint probability distribution represented by the very DHRF. In order to do that, an inference algorithm is applied, relying on Gibbs sampling as in regular HRFs [2]. Due to the particularities of the present model, inference takes place in two steps. The first step corresponds to the burn-in period. This means that each emission probability model (i.e., each HRF in the DHRF) is initialized at random according to a uniform distribution, and then subjected to an amount of Gibbs iterations that equals the burn-in time. The second step corresponds to the actual sampling phase. The algorithm aims at generating a number of sequences, and each one of them contains observations drawn from the emission probability distributions associated with different latent variables. Hence, for each sequence the initial state is selected and an observation (sampled from the corresponding HRF) is added to the sequence. Then, the next state is determined according to the transition probability distribution, and so on, until a final state is reached. The final state is eventually dismissed, after a certain number of iterations, according to its exit-probability. The core of the process is the generation of individual observations: an observation is sampled from the HRF corresponding to the current latent variable, and it is added to the sequence. Next, the algorithm prescribes to loop over the same latent variable, sampling from the same HRF, for as long as the *intra-sample time*. The latter parameter has to be tuned via model selection techniques, and it defines the expected number of iterations occurring between two samples so that they turn out to be stochastically independent. Then, the next latent variable is generated, and so on. In so doing, sampling from different HRFs is feasible, avoiding problems that may arise from the fact that the number of observations an HRF generates is not known a priori.

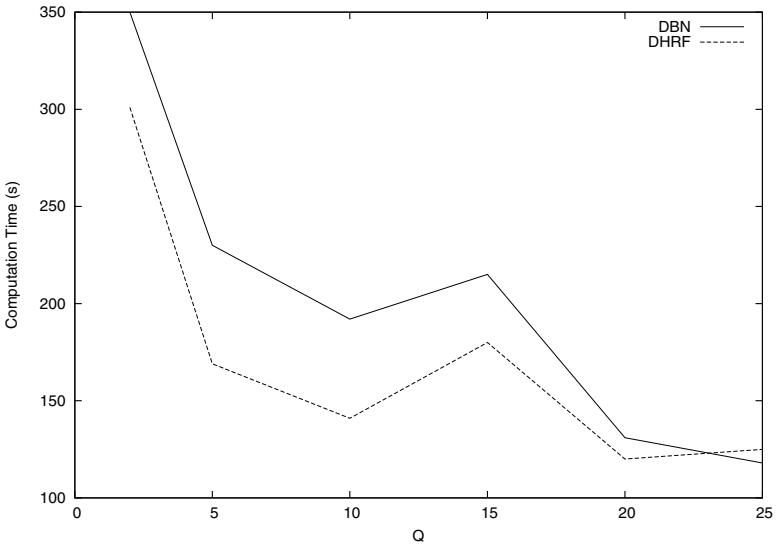

Fig. 1. Computational time of learning for DHRFs and DBNs, respectively, as functions of Q

3.2 Experiments

As we stated in the companion paper [5], the modeling capabilities of DHRFs subsume (by definition, due to the modeling capabilities of HRFs w.r.t. those of BNs) those of DBNs. To all practical ends, it is crucial to compare the performance of DHRFs and DBNs in terms of computational burden on the field. Extension to sequence modeling of the nice computational behavior of static HRFs is one of the main motivations behind the very development of DHRFs in the first place [5]. The focus is thence on the behavior of the learning time with respect to the variations of sensible parameters. These parameters are: (i) the number Q of latent random variables in the model; (ii) the number n of observable random variables in the model. Moreover, statistical (likelihood ratio-like) tests are applied, allowing for a comparison of the fitness of the joint densities estimated via DHRFs and DBNs. The corresponding results are presented hereafter (sections 3.3 and 3.4, respectively). Two different probability distributions, or "classes" ω_1 and ω_2 were generated and used for training/testing the learning machines. In practice, the data drawn from ω_1 and ω_2 were generated by separate generative DHRFs first, and later learned by two class-specific graphical models. In the following, all the results that are not presented in a class-specific fashion are to be intended as overall results, obtained on the whole data sample (inclusive of data drawn from both ω_1 and ω_2). Throughout all the simulations, data generation took

place as stated in section 3.1. The data embraced: (i) a training set of S sequences for each class; (ii) a validation set of $S/10$ sequences for each class; (iii) a test set of $S/5$ sequences for each class. In all the simulations we used observable variables whose discrete definition domains had cardinality ranging from 2 to 6 (at random). The experiments were carried out on a computer having PC architecture with CPU Intel Core i5-460M (2.53GHz) and 4GB of RAM.

3.3 Varying the Number of Latent Variables

Here we present a study of the performances of learning algorithms with respect to variations in the number Q of latent variables in the model (i.e. the number of HRFs to be learned). In this scenario we set $n = 20$, $S = 150$, and all sequences have randomly variable length (namely, roughly 100 observations long on average). Remember that in DMBM each HRF H_q is learned on a dataset D_q, obtained via Viterbi segmentation of the sequences in the training set D, as stated in section 2. As a consequence, if Q increases then the expected cardinality of each D_q (for $q = 1, \ldots, Q$) decreases. Hence, our first concern is investigating whether the problem of learning a higher number Q of HRFs leads to an increase in the computation time, or the latter is counterbalanced by the fact that the HRFs have to learn from smaller datasets D_q. This will throw a light on the efficiency of the divide and conquer philosophy underlying DHRFs, a strategy in line with the nature of the bare HRF which turned out to be thriving. Secondly, a direct comparison with DBNs is sought, and accomplished thenceforth.

Table 1. Pseudo log-likelihood with respect to Q. DHRF $log(P^*)$-t and DHRF $log(P^*)$-D are the pseudo log-likelihoods yielded by the DHRF on the test set and on the training set, respectively. DBN $log(P^*)$-t is the pseudo log-likelihoods yielded by the DBN on the test set. The DHRF ratio is the value $(log(P^*) - t)/(log(P^* - D))$.

Q	DHRF $log(P^*) - t$	DHRF $log(P^*) - D$	DHRF ratio	DBN $log(P^*) - t$
2	-62916	-369528	5,87	-74777
5	-68754	-356400	5,18	-79358
10	-76196	-349941	4,59	-83293
15	-86127	-412663	4,79	-94502
20	-74013	-366310	4,94	-78684
25	-77350	-378012	4,88	-80249

The results of the simulations are shown in figure 1. It is seen that, as Q grows, the computational burden of DHRFs is relatively stable. Yet, its predominant trend is even slowly decreasing. This suggests the preliminary conclusion that the computational complexity caused by the introduction of additional latent variables is indeed counterbalanced by the proportional decrease in the cardinality of the segmented datasets D_q. This consideration may encourage the

practitioner to apply DHRFs including a large number of latent variables, since an increase in expressiveness of the model may be expected without any loss in terms of complexity. Unfortunately, this may not be taken as a general rule of the thumb. In fact, as shown by the values of the *DHRF ratio* in table 1, an increase of Q over a certain threshold leads to a poor generalization capability. The DHRF ratio is defined as $(log(P^*)-t)/(log(P^*)-D)$, that is clearly an expression of the generalization capability of the learning machine. Of course, this phenomenon is just an instance, in the present scenario, of the traditional bias-variance dilemma (or, of the over-learning occurring as a consequence of the implicit increase in the Vapnik-Chervonenkis dimension of the resulting DHRF). Furthermore, in case the forward-backward algorithm for parameter learning [5] is applied afterwards, the overall time complexity would be affected by Q eventually (since in that variant of the algorithm the dataset D does not undergo initial segmentation, so that each one of the Q HRFs in the DHRF would learn from the whole D). Established model selection techniques are thus required from time to time in order to come up with a value for Q which fits the data at hand.

Finally, it is noteworthy that the results shown in table 1 highlight the significant difference between the $log(P^*)$ yielded by the DHRF and by the DBN, respectively. The former model is able to capture more statistical information (i.e., more conditional (in)dependences) from the training data than the latter one in a (at least) tantamount time, as expected.

3.4 Varying the Number of Observable Variables

Next, we analyze the performance of the learning algorithms with respect to variations in the number n of observable variables. Here we set $Q = 5$, $S = 200$, and random, variable sequence length (roughly 50 observations per sequence, on average). The present experiment is of the utmost relevance to our goals, since n is the most sensible parameter in relation to both the computational efficiency of learning in graphical models, and the capability to fit real-world scenarios that involve a number of random quantities (or, features). Studies presented in [3,4] show that the HRF scales very well with respect to n, while generally BNs and Markov random fields do not. As for DHRFs for sequences, the results of the simulations are reported in tables 2 and 3. The computational efficiency of DHRFs with DMBM in modeling large sets of random variables is immediately seen from figure 2. Table 2 reports also on the experiments we carried out with $n = 80$ and $n = 100$, where DBNs could not complete their learning process within an acceptable deal of time (i.e., the corresponding experiments had to be terminated). Basically, the computation time appears to increase (with n) in an approximately linear way, while the complexity of the learning algorithms for DBNs increase roughly exponentially.

Next, we investigate the variations of the pseudo log-likelihood $log(P^*)$ yielded by the two models on the test set. As shown in figure 3, the DHRF scores significantly higher than the DBN. Table 3 reports on the corresponding values

Table 2. Learning time in minutes and seconds (min:sec) as a function of n. The values for $n = 80$ and $n = 100$ could not be computed for the DBN.

n	DHRF time	DBN time
10	2:15	0:39
20	3:26	5:07
30	6:59	31:28
40	5:46	83:34
50	9:09	192:57
60	15:12	712:10
80	21:45	-
100	27:57	-

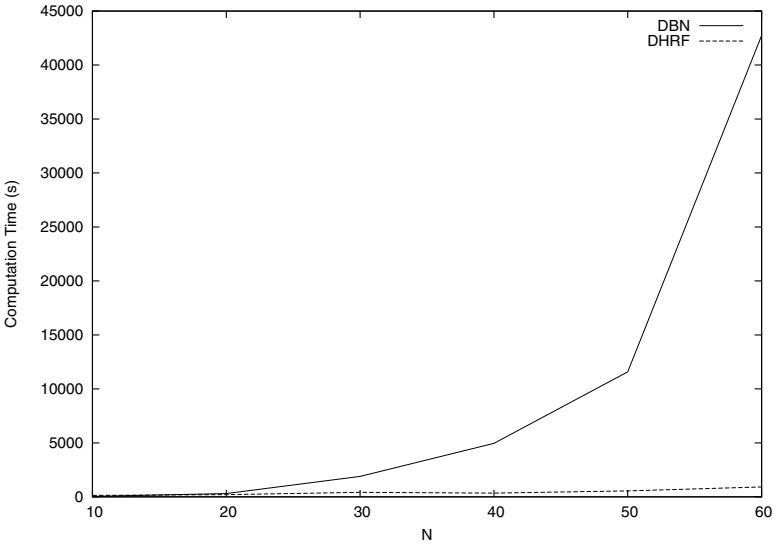

Fig. 2. Graphical comparison of the learning time for DHRF and DBN, respectively, as functions of n

Table 3. Pseudo likelihood ratio (PL-ratio) obtained over the data distributions ω_1 and ω_2 as a function of n

n	PL-ratio for ω_1	PL-ratio for ω_2
10	4834	13042
20	10472	16118
30	21394	29286
40	22708	17896
50	33730	29080
60	50916	36454

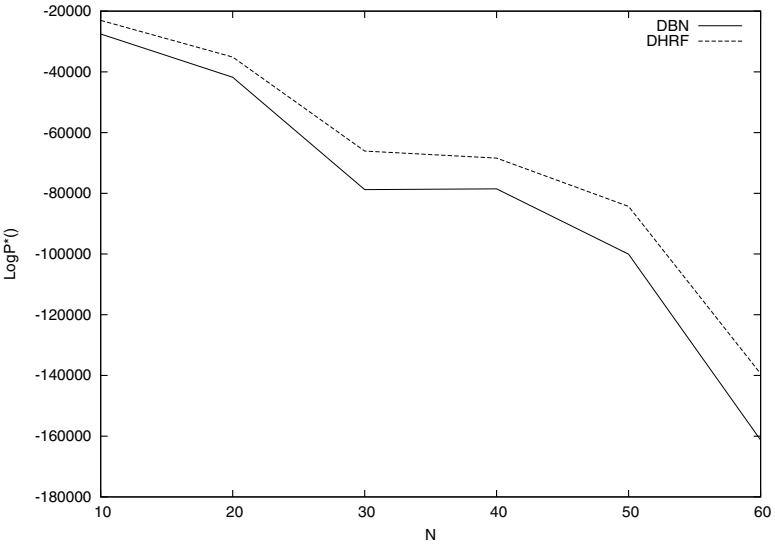

Fig. 3. Graphical comparison of the pseudo log-likelihoods yielded by the two models as functions of n

of pseudo likelihood ratio (PL-ratio) between the two models for the separate populations ω_1 and ω_2, respectively. The PL-ratio is defined as:

$$\begin{aligned} PL - ratio &= -2ln\left\{\frac{P^*_{DBN}(Testset)}{P^*_{DHRF}(Testset)}\right\} \\ &= -2ln(P^*_{DBN}(Testset)) + 2ln(P^*_{DHRF}(Testset)) \end{aligned} \tag{2}$$

This is a proper instance of the classic likelihood-ratio statistical test. In practice, positive values of the pseudo likelihood ratio entail the statistically-grounded fact that the DHRF fits the data better than the DBN does. As seen from table 3, we observed $PL - ratio >> 0$ in a systematic manner throughout the simulations. Put in words, the experimental evidence shows that (in the present scenario) the DHRF exhibits an improved behavior over the DBN both in terms of the capability to fit the data, and of the corresponding computational burden.

4 Conclusion

The paper introduced a technique for structure learning in DHRFs. The technique turns out to be a viable method for capturing the statistical (in)dependencies among the random variables within a sequence. Complexity issues were tackled by means of adequate strategies from classic literature on probabilistic graphical models. Performance of the resulting learning machine, suitable for the classification of sequential patterns, was empirically evaluated

in a synthetic task. Preliminary results show that DHRFs compare favorably with DBNs in terms of computational burden, with no loss in terms of modeling capability.

Acknowledgments. The invaluable support by Antonino Freno is gratefully acknowledged.

References

1. Freno, A.: JProGraM - PRObabilistic GRAphical Models in Java (2009), `http://www.dii.unisi.it/~freno/JProGraM.html`
2. Freno, A., Trentin, E.: Hybrid Random Fields: A Scalable Approach to Structure and Parameter Learning in Probabilistic Graphical Models. Springer (2011)
3. Freno, A., Trentin, E., Gori, M.: Scalable Pseudo-Likelihood Estimation in Hybrid Random Fields. In: Elder, J.F., Fogelman-Souli, F., Flach, P., Zaki, M. (eds.) Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 319–327. ACM (2009)
4. Freno, A., Trentin, E., Gori, M.: Scalable Statistical Learning: A Modular Bayesian/Markov Network Approach. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2009), pp. 890–897. IEEE (2009)
5. Trentin, E., Bongini, M.: Towards a Novel Probabilistic Graphical Model of Sequential Data: Fundamental Notions and a Solution to the Problem of Parameter Learning. In: Mana, N., Schwenker, F., Trentin, E. (eds.) ANNPR 2012. LNCS (LNAI), vol. 7477, pp. 72–81. Springer, Heidelberg (2012)