SUPERVISED TRAINING OF CONVERSIVE HIDDEN NON-MARKOVIAN MODELS: INCREASING USABILITY FOR GESTURE RECOGNITION

Sascha Bosse[(a)], Claudia Krull[(b)], Graham Horton[(c)]

[(a)(b)(c)] Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg, Germany

[(a)]sascha.bosse@ovgu.de, [(b)]claudia.krull@ovgu.de, [(c)]graham.horton@ovgu.de

ABSTRACT

Hidden non-Markovian Models (HnMMs) were introduced and formalized as an extension of Hidden Markov Models for the analysis of partially observable stochastic processes. Their main advantage over HMM is the possibility to model arbitrary distributions for state transition duration, so that the unobservable stochastic process needs not to be Markovian. Besides academic examples, HnMMs were applied to gesture recognition and performed well in distinguishing similar gestures with different execution speeds. While the Proxel-Method enabled the evaluation for arbitrary HnMMs, there was no opportunity to train these models. Therefore, the models for different gestures had to be parameterized manually. This fact reduced the applicability in real gesture recognition dramatically. This paper presents a solution to this problem, introducing a supervised training approach that increases the applicability of HnMMs in gesture recognition.

1. INTRODUCTION

Hidden non-Markovian Models (HnMMs) were introduced and formalized by Krull and Horton (2009) as an extension of Hidden Markov Models for the analysis of partially observable stochastic processes. Their main advantage over HMM is the possibility to model arbitrary distributions for state transition durations, so that the unobservable stochastic process needs not to be Markovian.

Besides academic examples (e.g. Buchholz et. al 2010; Krull et. al 2010), HnMMs were applied to gesture recognition and performed well in distinguishing similar gestures with different execution speeds (Bosse et al. 2011). For that purpose, significant changes in gesture acceleration were logged while execution and the likelihood of different HnMMs to generate such a sequence was computed. The model with the highest value in this evaluation task represents the recognized gesture.

While the Proxel-Method developed by Horton (2002) enabled the evaluation for arbitrary HnMMs, there was no opportunity to train these models automatically. Therefore, the models for different gestures had to be parameterized manually. This fact reduced the applicability in real gesture recognition dramatically.

This paper has the goal to present a solution to this problem, introducing a supervised training approach that increases the applicability of HnMMs in particular in gesture recognition, but also in other application areas where a fully specified model of the hidden system is not readily available.

The next section will review some existing training methods of related paradigms and introduce HnMMs. The third section describes the steps of the training approach and the fourth section comments on implementation details. The experiments section contains two test cases. The paper is concluded by the sixth section, which evaluates the approach presented and highlights some areas future work.

2. RELATED WORK

Training a mathematical model to increase its applicability is a well addressed problem in *Machine Learning*. There are two basic forms of Machine Learning: Supervised and unsupervised learning. In supervised learning input and desired output data is used, so that the model can learn the relationship between them (e.g. Classification). In unsupervised learning, no output data is given and the model has to describe the distribution of the data (e.g. Clustering) (Marsland 2009).

For Hidden Markov Models there exist well known unsupervised training algorithms, like Baum-Welch- and Viterbi-Training (Fink 2008). These methods are iterative and guarantee a greater or equal likelihood of the model after each iteration. Buchholz (2012) adapted the Baum-Welch-Algorithm to a subclass of HnMMs also used in gesture recognition, but there is still missing a concept to train arbitrarily distributed state transition durations.

On the other hand, approaches for supervised training of Hidden Markov Models like (Mamitsuka 1998) were developed to train the evaluation probability to a specific target value. An algorithm that trains a Hidden Markov Model from sequences with desired states could not be found in literature, probably because the computation of transition and output probabilities would be trivial in that case.

For the problem of gesture recognition addressed in (Bosse et. al 2011) none of these approaches is suitable in general. Because semantic information is encoded in the model, a training method must respect this and may not change the basic structure of the model. Because of this fact, the current state of the underlying model should be computed from training data, so that the training method can respect the semantics of the model.

2.1. Hidden non-Markovian Models

An HnMM after Krull and Horton (2009) consists of a state space with transitions between the states that are time-dependent. In a specific case of HnMMs (*Eall*), every transition must emit a symbol when it fires. Buchholz (2012) named this subclass Conversive HnMM and developed algorithms for all relevant problems. CHnMM assign every transition with a discrete random variable that indicates what emission probability each output symbol has. In addition, a distribution of the initial probability of each state is given.

The evaluation task of CHnMM is solved by an approach very similar to the Forward Algorithm from HMM (Buchholz 2012). This algorithm requires a completely defined system description, including the continuous distributions of transition firing times and discrete distributions for symbol emissions. These distributions must be provided by a training algorithm.

The state space of an HnMM can be computed from arbitrary models that represent discrete stochastic processes. Furthermore, the state space of a model does not need to be computed a-priori, so the training algorithm can parameterize a discrete stochastic model to avoid problems like state space explosion. In this work, augmented stochastic Petri nets are used as the user model.

2.2. Augmented Stochastic Petri Nets

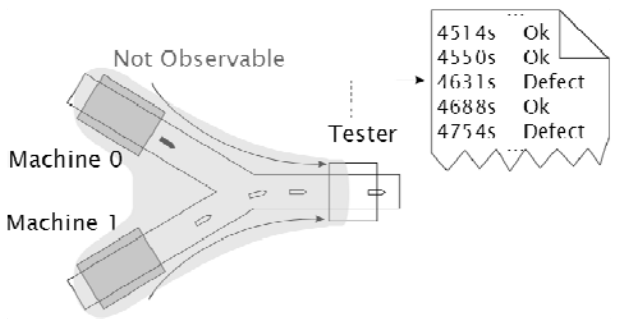
Krull (2009) defines Stochastic Petri Nets as a 7-tuple . is the set of places, a set of transitions and a set of arcs between places and transitions. must form a bipartite graph. is a set of so called inhibitor arcs while is a function that assigns integer values to each type of arc. is the initial marking of all places and can assign a so called guard function to each transition that indicates whether a transition is activated or deactivated for a specific marking.

A transition is activated if and only if every place connected to this transition has at least the number of tokens the connecting input arcs are assigned, no source of an inhibitor arc to this transition has at least the specified number of tokens and the corresponding guard function of this transition evaluates to true in the current marking. After a randomly distributed amount of time, the transition fires, deleting the required tokens and creating tokens in the places the transition is connected to. If another activated transition fires before that time and the new marking disables the transition, there are two policies: If the transition is marked as “Race Age”, the transition saves the remaining firing time and resumes the countdown when it is active again. With the policy “Race Enabled” the whole activation time is deleted and upon re-enabling, the firing distribution is sampled again.

Buchholz (2012) defines augmented stochastic Petri nets. Here each transition is augmented by the symbol emissions it can produce with probabilities for each possible symbol. The state space of such an ASPN is an HnMM. ASPN can be considered the user model corresponding to the computational model HnMM.

2.3. Example System Description

The current paper is illustrated using an example first defined by Buchholz et. al (2010). Two machines process products in randomly distributed intervals. These products are tested after both machine results are joined. The tester produces a protocol with test timestamp and state of the product (*OK* or *Defect*). Figure 1 illustrates the system and shows an example of such a system protocol.



```
...
4514s  Ok
4550s  Ok
4631s  Defect
4688s  Ok
4754s  Defect
...
```

Figure 1: Schematic of tester example

This system can be converted into the augmented stochastic Petri net (Buchholz 2012) shown in Figure 2. The ASPN represents the system as a stochastic Petri net with output symbols representing the tester results attached to the state transitions.

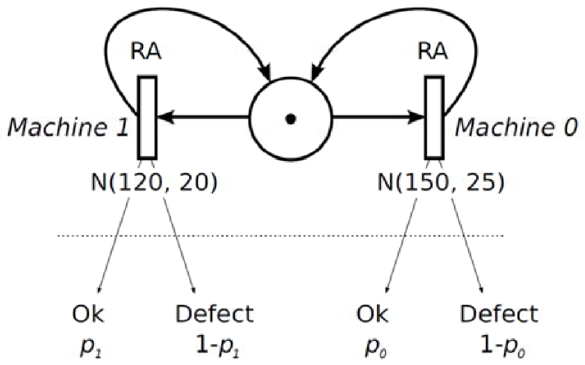
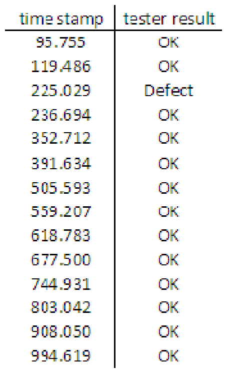


Figure 2: Augmented stochastic Petri net of the tester example



time stamp	tester result
95.755	OK
119.486	OK
225.029	Defect
236.694	OK
352.712	OK
391.634	OK
505.593	OK
559.207	OK
618.783	OK
677.500	OK
744.931	OK
803.042	OK
908.050	OK
994.619	OK

Figure 3: Example protocol of the tester system

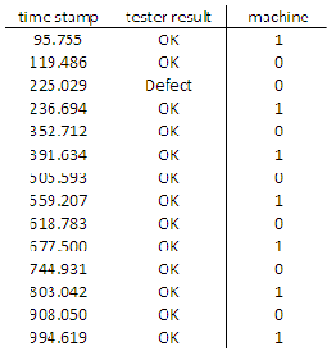
The protocol produced by the tester (see Figure 3 for an example) does not contain information on the machine that processed a particular product, e.g. produced a certain defective item. Therefore, this part of the model can be considered unobservable. The task to be solved using HnMM is therefore to reassign the different protocol entries to the machines, thereby determining the sources of defective items.

3. APPROACH

The goal of the desired approach is a supervised training of a Hidden non-Markovian Model to improve the recognition accuracy for a given real system. For that purpose, the hidden discrete model must be adapted by using training data that contains more information than the data from the real system to be reconstructed later on. A second property of the approach should be that the trained model gets nearer to the original with increasing training data amount.

3.1. Preconditions

A symbol sequence entry of an HnMM consists of two parts: timestamp and symbol. For training purposes, those sequence entries are annotated with the transition that generated the symbol emission as shown in Figure 4. The user model to be trained is an ASPN with a known structure. The model parameters that will be trained are the continuous distribution functions of the timed transition and the corresponding symbol output probabilities. An initial state probability distribution will also be determined. This corresponds to parametric training.



time stamp	tester result	machine
95.755	OK	1
119.486	OK	0
225.029	Defect	0
236.694	OK	1
352.712	OK	0
391.634	OK	1
505.593	OK	0
559.207	OK	1
618.783	OK	0
677.500	OK	1
744.931	OK	0
803.042	OK	1
908.050	OK	0
994.619	OK	1

Figure 4: Annotated example trace of the tester system

Two main tasks can be identified to solve this problem: Firstly, for every transition samples of the firing time must be computed from the annotated symbol sequence and secondly, the distribution must be estimated from these firing time samples.

3.2. Computation of firing time samples

For every event in the protocol at time representing a firing transition, the corresponding firing time can be computed in the following way:

(1)

That means the relative firing t_{fire} time is the difference between the timestamp of the firing t and the time when the transition was last activated t_{act}. is the current age of the transition which is not equal to zero if the transition is race age and was activated before but did not fire. While the timestamp is available of course, the other two values are not from the protocol itself.

But they can be computed easily when the protocol is processed sequentially. Given the initial marking and the structure of the underlying model, at every timestamp the marking - depending on the given transition - is stored. From this marking the activated transitions can be inferred. For every transition this activation time and for all race age transitions that are not longer activated the age, i.e. the difference between timestamp and last activation time, is stored.

With this procedure, every entry in the sequence produces a relative firing time for the given transition, so that a collection of firing times arises.

3.3. Estimating the Probability Distribution

Estimating a probability distribution from a set of realizations is a problem well addressed in density estimation. The methods for this task are divided in parametric and non-parametric methods (Eggermont and LaRiccia 2001). The first mentioned is about estimating the parameters of known distribution types, the normal distribution for example. If none of these distributions fits the data, a non-parametric method can be applied. This can be a simple histogram or the more complex kernel estimation.

4. IMPLEMENTATION

To test the supervised training, the approach needed to be implemented. The computation of firing times can be done with a Petri net simulation providing the following methods: For a specific marking, a list of active transitions and for a given transition, a new marking must be returned. In addition to that, some auxiliary variables are needed. Besides the current index corresponding to the processed symbol sequence entry, the current marking of the Petri net must be saved (integer array, size). Also for every transition, the last activation times and the age times must be stored (float array, size). The latter array can be minimized, if only race age transitions are considered.

In addition to that, for every transition the symbol emissions must be counted.

From the collected data, the distributions must be computed. The estimation of the discrete probability distribution of the symbol emissions are easy to compute. Estimator is the relative frequency of the symbol emission.

For the estimation of the continuous probability distributions of the firing times the kernel estimation seems to be the most general approach. Although it can be shown that the error of this method tends to zero with enough data (Devroye and Lugosi 2001), a parametric estimation reduces the error much faster with respect to training data if the true distribution is a known one. This holds because less information is need to parameterize a known distribution than an unknown one.

Due to these facts, both approaches are considered in this concrete implementation. Firstly, an optimization for some known probability densities is carried out. Minimizing the square error between observed values and expected values in defined intervals returns the distribution fitting best.

If this distribution does not pass a chi-square-test at a specific significance level (from Banks et. al 2001), kernel estimation is performed with Gauss kernel and window size chosen according to Silverman (1998).

The needed cumulative distribution function is computed symbolically for the corresponding densities and through numerical integration for densities without symbolic integrations and kernel density.

With the estimated distributions, the augmented stochastic Petri net can be parameterized and the CHnMM evaluation algorithm is now able to compute the evaluation probabilities of other protocol sequences.

5. EXPERIMENTS

To test the developed approach, two experiments are performed. Since it is not clear yet, how to generate training data from real systems, training is tested using two academic models, where a simulation model is available to generate both training data and test sequences.

Therefore we have as reference value to estimate the quality of the trained models the evaluation probability of each sequence computed using the actual generating model. The training approach is successful if the evaluation probability of the sequence for the trained model is similar to this reference value. In particular, the difference between the values should decrease the more training data is available.

To illustrate the different effects of parametric and non-parametric estimation, both approaches are shown in the experiment results.

The computation time needed for extracting and finding the distributions from the data, as well as the runtime of the evaluation task were both under one minute and are therefore not considered in this paper.

5.1. Tester Example

The first experiment is carried out using the tester example system described in Section 2.3. For training purposes, augmented training protocols of different lengths were generated (20, 50, 100, 200, 500, 1,000, 2,500 and 10,000 entries for each transition). One model is trained using kernel estimation, and the parametric training is performed using normal distributions.

Then for ten different protocols, the evaluation probability of the protocol given the trained model is computed. Reference evaluation probabilities are computed using the CHnMM of the generating model. The mean relative difference (can also be interpreted as the error) of the trained model probabilities and the reference values using a particular amount of training data was computed using Equation (2).

$$\overline{Err_n} = \frac{1}{10}\sum_{i=1}^{10}\frac{|p_{i,n}-p_{i,ref}|}{p_{i,ref}} \qquad (2)$$

The reference evaluation probability of test sequence i is given by $p_{i,ref}$ while the evaluation probability of test sequence i obtained from a model trained with n training samples is given by $p_{i,n}$. The development of this error value with increasing training data amount is presented in Figure 5.

The first observation is that increasing the training data amount decreases the error and therefore increases the quality of the model, which is a necessary feature of a training approach. Because of the stochastic nature of the training data this decrease is not smooth for little training data. The effect is more smooth when more training date is available, which suggests that a minimum amount of training data needs to be available to obtain useful results.

The error decreases faster when training normal distributions, since the data was generated using a model with normal distributions. The models containing kernel estimations of the distributions retain a significantly higher error in the evaluation probability with the same amount of training data.

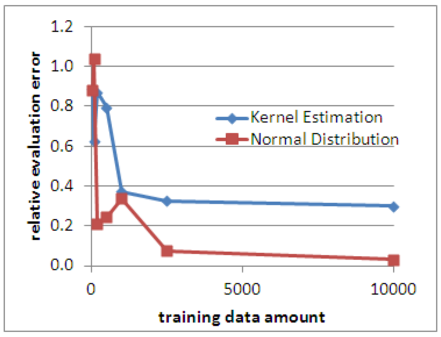


Figure 5: Mean relative difference of evaluation probabilities depending on training data amount

5.2. Non-Parametric Motion Sensor Example

In real applications, data is often assumed to meet a parametric distribution. But that is not always the case. Therefore, a second experiment was performed using a model with more complex distributions. The example is a very simple motion sensor in a movable utility (e.g. game pad of a games console, pen or wiper on smart board). The task of the model would be to distinguish between deliberate movements the user and random influences on the utility such as jitter, jolting or draft.

The given system structure is very simple as shown in Figure 6 in the form of an ASPN. The two places of the ASPN represent the states *Idle* and *Busy*. *Idle* meaning that the utility is not in motion and *Busy* meaning that it is being used, and the corresponding motions should be registered. A speed threshold has been defined in order to distinguish between deliberate movements and random influences. The transitions between the states should cause the speed value to rise above (*Begin Moving*) or fall below (*Seize Moving*) the given threshold.

To emulate non-parametric data, transition *Begin Moving* has a firing time that is a combination of two normal distributions, N(60,10) and N(75,15), where a random process picks each of the distributions with equal probability and then samples that one. The firing time of transition *Seize Moving* is a convolution of a normal distribution and an exponential distribution, N(4,0.5) and Exp(0.5).

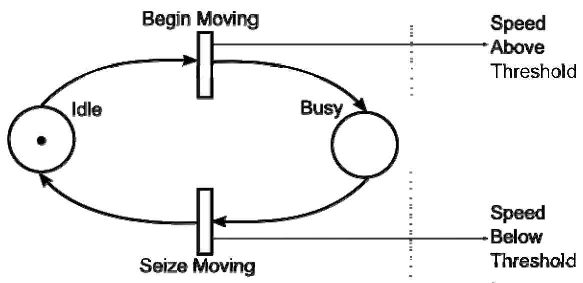


Figure 6: Augmented stochastic Petri net of non-parametric motion sensor model

For this model, training sequences of different length were generated using a discrete event-based simulation (20, 50, 100, 200, 500, 1,000, 2,500, 5,000 and 10,000 fire times for each transition). Additionally, ten test sequences were generated. As reference values for the evaluation probabilities the results of a model trained with almost 14000 items of data were used, because the non-parametric distributions could not be used directly in the analysis method. A generation of training sequences from a real motion sensor would require the users to indicate the start and end of deliberate movements e.g. by pressing a button on the game pad.

As estimate of the quality of the trained model, we again computed the mean relative error for different amounts of training data using Equation (2). The development of the error for the trained models containing kernel estimations as distributions is shown in Figure 7. It contains the mean error value as well as the maximum and minimum value obtained of the ten test sequences. The error development using parametric estimators can be seen in Figure 8 using a different scale than in the non-parametric case.

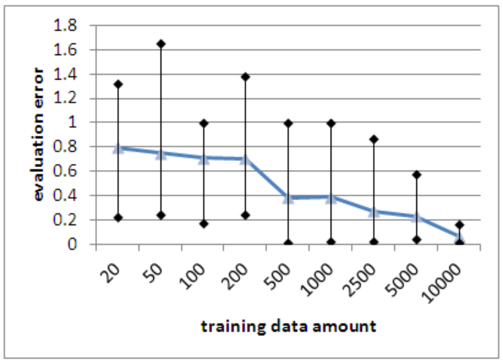


Figure 7: Error of evaluation probabilities depending on training data amount for non-parametric estimator

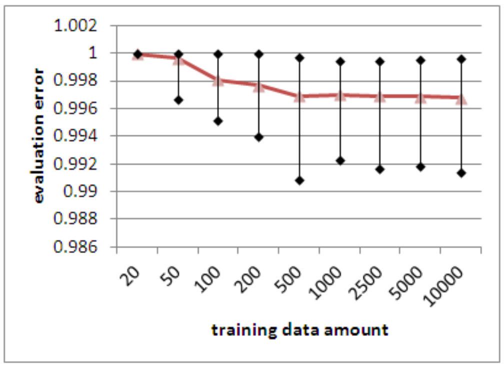


Figure 8: Error of evaluation probabilities depending on training data amount for parametric estimator

The error using the parametric estimator does not decrease significantly with increasing training data amount, while the non-parametric estimator performs significantly better. This is to be expected, since the original distributions were designed not to fit a known distribution function.

Maybe a more complicated parametric approach could explain the data better than the here estimated Erlang distribution for T1 and lognormal distribution for T2, but that was not within the scope of the experiment.

5.3. Experiment Discussion

The experiments show that a supervised training of Hidden non-Markovian Models is possible, and that the approach is capable of training parametric and non-parametric distributions. The quality of the trained models, in terms of difference in evaluation probability to a reference model, increases with increasing training data amount, when a certain minimal amount of data is available.

It becomes clear that parametric estimation yields better results even with little training data if the original data corresponds to a known distribution function. If this is not the case, a non-parametric estimator is a good choice because it always tends to the original data distribution given enough training examples.

6. CONCLUSION

This paper presented a supervised training approach for Hidden non-Markovian Models. Even though it is based on existing methods, only their combination leads to a first successful attempt to train Hidden non-Markovian Model. Experiments showed that this approach is able to minimize the gap between the source system and the trained model if the model structure is sufficient to map this system and enough training data is available.

The method to collect firing times of specific transitions is applicable to any augmented stochastic Petri net described as above. Parametric and non-parametric estimators are capable of approximating arbitrary data distributions. If the data fits to a parametric distribution the corresponding estimators reach that goal much faster than the non-parametric ones, although the latter approach is more general.

Disadvantages of the approach arise from the preconditions. An augmented Petri net fully specified in its structure is needed, otherwise the method to collect firing times cannot be applied. In addition to that this supervised approach also needs information on which transition has fired for each protocol entry. That means that the unobservable system part must be observable for the generation of training data. This might be a key problem for the applicability of the approach. However, in the case of gesture recognition within the approach of Bosse et. al (2011), it can be applied. For that purpose, users will need to mark their sequence of data generated from movements with the desired gesture phases.

In future work the approach should be applied to real gesture recognition with Hidden non-Markovian Models. This task involves scenarios with several users to collect training data from their specific gesticulation. Later on the users should carry out the defined gestures to see if the recognition rate increases through training.

A second opportunity of research is the applicability of the approach to so-called normal HnMMs in which not every transition does emit a symbol. That means the current state of the HnMM respectively the current marking of the Petri net when generating the firing times may be non-deterministic.

REFERENCES

Banks, J., Carson, J.S. II, Nelson, B.L., Nicol, D.M., 2001. Discrete-Event System Simulation. Upper Saddle River, NJ: Prentice-Hall

Bosse, S., Krull, C., Horton, G., 2011. MODELING OF GESTURES WITH DIFFERING EXECUTION SPEEDS: Are Hidden non-Markovian Models Applicable for Gesture Recognition, 10th International Conference on Modelling & Applied Simulation (MAS). September 2011, Rome, Italy.

Buchholz, R., Krull, C., Strigl, T., Horton, G., 2010. Using Hidden non-Markovian Models to Reconstruct System Behavior in Partially-Observable Systems, 3rd International Conference on Simulation Tools and Techniques, March 2010, Torremonlinos, Spain.

Buchholz, R., 2012. Conversive Hidden non-Markovian Models. Doctoral Thesis, Otto-von-Guericke-University Magdeburg, Germany.

Devroye, L, Lugosi, G., 2001. Combinatorial Methods in Density Estimation. New York: Springer-Verlag.

Eggermont, P.P.B., LaRiccia, V.N, 2001. Maximum Penalized Likelihood Estimation. New York: Springer-Verlag.

Fink, G.A., 2008. Markov Models for Pattern Recognition. Berlin: Springer-Verlag

Horton, G., 2002. A new Paradigm for the Numerical Simulation of Stochastic Petri Nets with General Firing Times, 14th European Simulation Symposium. October 2002, Dresden, Germany.

Krull, C., 2008. Discrete-Time Markov Chains: Advanced Applications in Simulation. Doctoral Thesis, Otto-von-Guericke-University Magdeburg, Germany.

Krull, C., Horton, G., 2009. HIDDEN NON-MARKOVIAN MODELS: FORMALIZATION AND SOLUTION APPROACHES, 6th Vienna International Conference on Mathematical Modeling. February 2009, Vienna, Austria.

Krull, C., Buchholz, R., Horton, G., 2010. MATCHING HIDDEN NON-MARKOVIAN MODELS: DIAGNOSING ILLNESSES BASED ON RECORDED SYMPTOMS, 24th European Simulation and Modeling Conference, October 2010, Hasselt, Belgium.

Mamitsuka, H., 1998. Predictiong Peptides that bind to MHC molecules using Supervised Learning of Hidden Markov Models, PROTEINS: Structure, Function and Genetics, Vol. 33, 460-474

Marsland, S., 2009. Machine Learning: An Algorithmic Perspective. Boca Raton, FL: CRC Press

Silverman, B.W., 1998. DENSITY ESTIMATION FOR STATISTICS AND DATA ANALYSIS, Monographs on Statistics and Applied Probabilty. London: Chapman and Hall.

Wickborn, F., Horton, G., Heller, S., Engelhard, F., 2005. A General-Purpose Proxel Simulator for an Industrial Software Tool, 18th Symposium of Simulation Techniques. September 2005, Erlangen, Germany