BAYESIAN LEARNING OF LOGICAL HIDDEN MARKOV MODELS

T. Raiko[1,2] *K. Kersting*[1] *J. Karhunen*[2]
L. De Raedt[1]

[1]Institute for Computer Science
Machine Learning Lab
Albert-Ludwigs University of Freiburg
Georges-Koehler-Allee, Building 079
79112 Freiburg, Germany

[2]Helsinki University of Technology
Laboratory of Computer and
Information Science,
P.O. Box 5400,
02015 HUT, Finland

Abstract

Logical hidden Markov models (LOHMMs) are a generalisation of hidden Markov models to analyze sequences of logical atoms. Transitions are factorized into two steps, selecting an atom and instantiating the variables. Unification is used to share information among states, and between states and observations. In this paper, we show how LOHMMs can be learned using Bayesian methods. Some estimators are compared and parameter estimation is tested with synthetic data.

1 INTRODUCTION

Hidden Markov models [13] (HMMs) are among the most widely and successfully used tools for the analysis of sequential data. Areas of application include computational biology, user modeling, speech recognition, (stochastic) natural language processing, and robotics. The analyzed sequences consist of symbols that could be named as $a, b, c, \ldots$ In a logical sense, the alphabets are propositional. In case the propositional elements are replaced by logical atoms [11], we end up with sequences like $a(c), b(e, c), a(b)$. Logical sequences are a natural representation e.g. for the secondary structure of proteins [10] and Unix shell command logs.

If we would like to analyze logical sequences with a propositional hidden Markov model, the sequence has to be propositionalized by e.g. leaving the arguments out and getting a, b, a or flattening the structure getting a_c, b_{ec}, a_b. In the former case, we lose a lot of information. In the latter case, the number of parameters becomes very high and we lose the shared information between $a(c)$ and $a(b)$. Logical hidden Markov models [10] (LOHMMs), overcome these weaknesses. In this paper, we go into details on how to learn a LOHMM from data.

This paper is organized as follows. In the next Section, we will describe logical hidden Markov models. In Section 3, we discuss learning them in general from Bayesian point of view. Section 4 describes the adaptation of the Baum-Welch re-estimation in more detail. Section 5 shows some synthetic experiments. Subsequently, we list related work.

2 LOGICAL HIDDEN MARKOV MODELS

This section gives a short introduction to logical hidden Markov models. A more thorough introduction can be found in [9].

A logical hidden Markov model (LOHMM) is a generative model, which can be described as a set of transitions and a set of selection probabilities for variables. For example

$$\begin{array}{rlrl} b(X,Y) \xleftarrow{0.7} start. & & b(X,Y) \xleftarrow{0.2:h(X)} b(X,Y). \\ b(X,X) \xleftarrow{0.3} start. & & b(Y,X) \xleftarrow{0.3:h(X)} b(X,Y). \\ & & b(X,Y) \xleftarrow{0.3:t(X)} b(X,Y). \\ b(X, blue) \xleftarrow{1.0:h(green)} b(red, blue). & & b(X,Z) \xleftarrow{0.2:t(X)} b(X,Y). \end{array} \tag{1}$$

could be interpreted as follows: The hidden state b has two arguments, the colours of the left and right ball. In the beginning, it is more probable to have same colours than by selecting independently. At each time step, a coin is thrown. The observation contains the result of the coin throw (h or t) and the colour of the left ball. If we observe a heads, it is possible that the balls are switched. If we observe a tail, the right ball might be replaced by a new one. There is one exception: if the left ball is red and the right one is blue, the observation is a heads with green, and the left ball is replaced.

Each transition includes 1) the logical atom called *body* which is the source of the transition; 2) the logical atom called *head* which is the destination of the transition; 3) the logical atom which is observed[1] and 4) the probability associated to the transition. The names body and head and the direction to the left are used since transitions of a LOHMM can be seen as augmented logical clauses.

For reasons that will become clear, we need the *more specific* relation among atoms. An atom a is (strictly) more specific than atom b if b can be transformed into a by substituting some of its variables, but a cannot be transformed into b similarly. For example $a(X,X)$ is more specific than $a(Y,Z)$ since the variables Y and Z can be substituted with X, but X cannot be substituted with both Y and Z at the same time.

A logical sequence can be generated from a LOHMM by repeating the steps below. At each time point, the process is in a (hidden) state represented by a logical atom. It starts in the special state called *start*.

1. The body, which is most specific for the current state, is selected (deterministically).

2. A transition is selected according to the transition probabilities.

3. Variables that appear in the body are instantiated to match the current state.

4. Other variables are instantiated using a selection probability μ.

5. The (instantiated) observation is added to the logical sequence.

[1]Transitions from *start* do not produce an observation.

6. The (instantiated) head is used as the current state and the LOHMM is left unchanged.

We use a simple selection distribution μ here: Each argument of an atom has a type and each type has a distribution over constants associated to it. In our example, the only type *colour* is defined as $\{0.4 : red, 0.3 : green, 0.3 : blue\}$.

A LOHMM is well-defined, if there is a unique most specific body for every state and if the transition probabilities for a single body sum up to one. For example, if there were bodies $b(X, X)$ and $b(X, red)$ in a LOHMM and the current state was $b(red, red)$, one could not say which body to use. The problem is avoided by adding transitions with the body $b(red, red)$.

LOHMMs combine logical reasoning with probabilistic (statistical) modelling. For example, if we know the LOHMM produced the sequence $h(green), h(blue)$, we can compute that the probability of being in the hidden state $b(green, blue)$ at time 1 is 0.43. If we also know, that the third observation is $h(green)$, we can be certain that the hidden state at time 1 was in fact $b(green, blue)$.

3 LEARNING A LOHMM FROM DATA

This section describes how a LOHMM can be learned from data using the Bayesian approach.

Let us first separate a LOHMM into its parameters $\boldsymbol{\theta}$ (the probabilities) and the structure $\mathcal{H}$ (the rest). Let us also assume a prior which generates a structure $\mathcal{H}$ with probability $p(\mathcal{H})$ and parameters for it with probability density $p(\boldsymbol{\theta} \mid \mathcal{H})$. Learning a single LOHMM $(\mathcal{H}, \boldsymbol{\theta})$ from data $\boldsymbol{X}$ corresponds to finding a good representative of the posterior probability mass:

$$p(\boldsymbol{\theta}, \mathcal{H} \mid \boldsymbol{X}) \propto p(\boldsymbol{X} \mid \boldsymbol{\theta}, \mathcal{H}) p(\boldsymbol{\theta} \mid \mathcal{H}) p(\mathcal{H}). \tag{2}$$

Instead of finding just one LOHMM, one could use an ensemble of them represented with a set of sampled points [5] or a distribution with a simple form [7]. This is out of scope of this paper.

Finding a good representative structure $\mathcal{H}$ involves a combinatorial search in the structure space, where for each structure candidate, the parameters $\boldsymbol{\theta}$ need to be estimated. One could first try structures that resemble good candidates by using inductive logic programming [12] techniques. Also, the information from other structures can be used to guide the parameter estimation. This is further research.

The representative parameters $\boldsymbol{\theta}$ for a given structure $\mathcal{H}$ can be found using estimation. There are two commonly used estimators: the maximum a posteriori (map) estimate $\boldsymbol{\theta}^{map}$ and the Bayes estimate $\boldsymbol{\theta}^B$. They are defined as

$$\boldsymbol{\theta}^{map} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{X} \mid \boldsymbol{\theta}, \mathcal{H}) p(\boldsymbol{\theta} \mid \mathcal{H}) \tag{3}$$

$$\boldsymbol{\theta}^B = \int p(\boldsymbol{X} \mid \boldsymbol{\theta}, \mathcal{H}) p(\boldsymbol{\theta} \mid \mathcal{H}) \boldsymbol{\theta} d\boldsymbol{\theta}. \tag{4}$$

The maximum likelihood estimator is a special case of the map estimator assuming the prior of the parameters $p(\boldsymbol{\theta} \mid \mathcal{H})$ to be uniform. Note that the Bayes estimate is unique whereas the map estimate is not.

One can also use the Bayes estimator componentwise (cB). Each component θ_k is estimated by

$$\theta_k^{cB} = \int p(\boldsymbol{X} \mid \boldsymbol{\theta}, \mathcal{H}) p(\boldsymbol{\theta} \mid \mathcal{H}) \theta_k d\theta_k \tag{5}$$

keeping all the other components constant. That is, $\boldsymbol{\theta}$ is $\boldsymbol{\theta}^{cB}$ with the kth component replaced by θ_k. The componentwise Bayes estimate is no longer unique.

4 PARAMETER ESTIMATION

The Baum-Welch algorithm[13] is used to estimate parameters of hidden Markov models from data. It is based on forward and backward procedures. These can be adapted for LOHMMs with few differences. The Baum-Welch algorithm is an instance of the expectation-maximisation (EM) algorithm. In the E-step, the expected values of the latent variables are estimated keeping the parameters constant, and in the M-step, the parameters are updated keeping the latent variables constant. The two steps are iterated until a convergence criterion is fulfilled. It has been shown [2] that the EM-algorithm converges to a fixed point. Let us now discuss these two steps.

4.1 Expectation step

The α and β values computed by the forward-backward algorithm are useful for many purposes. Firstly, the probability of the observation sequence given the LOHMM is simply $p_{seq} = \sum_{s \in S_T} \alpha_T(s) = \beta_0(start)$. Secondly, the probability of being in a state s at time t given the observation sequence is $\alpha_t(s)\beta_t(s)/p_{seq}$. Thirdly, computing the expected counts required for the maximization step, becomes simple.

The forward-backward algorithm is based on dynamic programming. To ease the computations, we find sets S_t in the beginning of learning. The set S_t is defined as the set of states at time t that are reachable from *start* and are not conflicting with the observation sequence $o_1, o_2, \ldots, o_T$. The reachable states can be found in two phases:

Collect Set $S_0 = \{start\}$. For each $t = 0, 1, \ldots, T$, using the states S_t as sources, find all states S_{t+1} where a transition can be made without conflict with the current observation o_t.

Prune For each $t = T, T-1, \ldots, 0$, remove state from S_t if there are only conflicting transitions to any of the states still in S_{t+1}.

The *start* state is pruned if and only if the LOHMM could not have produced the sequence.

Probabilities α and β are computed for each state in each S_t recursively. The $\alpha_t(s)$ is defined as the probability of the partial observation sequence $o_1, \ldots, o_{t-1}$ and state s at time t given the LOHMM. The $\beta_t(s)$ is the probability of the partial observation sequence $o_t, \ldots, o_T$ given state s at time t and the LOHMM. Set $\alpha_0(start) = 1.0$ and

$\beta_{T+1}(s) = 1.0$ for every $s \in S_{T+1}$. Recursion formulae are

$$\alpha_t(h) = \sum_{cl\in\mathcal{H}} \sum_{b\in S_{t-1}} \alpha_{t-1}(b) p_{cl} \mu \delta(cl, b, h, o_t) \tag{6}$$

$$\beta_t(b) = \sum_{cl\in\mathcal{H}} \sum_{h\in S_{t+1}} \beta_{t+1}(h) p_{cl} \mu \delta(cl, b, h, o_t), \tag{7}$$

where p_{cl} is the transition probability and μ is the selection probability of the instantiation of the new variables in the head and observation. The indicator function $\delta(cl, b, h, o_t) = 1$ whenever transition cl can take from state b to h observing o_t and the transition cl has the most specific body for b.

The expected counts for the maximisation step are computed using α and β values. Let $\nu(cl, b, h, t)$ be the probability of going from state b at time t to state h at time $t+1$ using the transition cl given the LOHMM and the observation sequence:

$$\nu(cl, b, h, t) = \frac{\alpha_t(b) p_{cl} \mu \beta_{t+1}(h) \delta(cl, b, h, o_t)}{p_{seq}}. \tag{8}$$

The expected count $s(cl)$ of using transition cl with the data set $\boldsymbol{X}$ is

$$s(cl) = \sum_{seq\in\boldsymbol{X}} \sum_{t=0}^{T_{seq}} \sum_{b\in S_t} \sum_{h\in S_{t+1}} \nu(cl, b, h, t), \tag{9}$$

where T_{seq} is the length of the sequence seq. The expected counts of the selection distribution μ are also obtained by summing terms ν analogously.

4.2 Maximisation step

Assuming the prior distribution of the parameters $p(\boldsymbol{\theta} \mid \mathcal{H})$ to be formed from Dirichlet distributions, makes the maximisation step easy. Let $\rho(j) \in [0, 1]$, $j = 1, \ldots, k$ be random variables such that $\sum_{j=1}^{k} \rho(j) = 1$. In our case, $\rho(j)$ are the transition probabilities p_{cl} corresponding to a certain body or the selection probabilities μ corresponding to a certain type. A Dirichlet distribution is defined as

$$P(\rho) = \prod_j \rho(j)^{z(j)-1}/C, \tag{10}$$

where $z(j)$, the pseudocounts, are parameters controlling the distribution and C is a normalisation constant, which guarantees that $\int P(\rho) d\rho = 1$. Note that a Dirichlet distribution with $\forall j, z(j) = 1$ is uniform.

The terms of log likelihood $\log p(\boldsymbol{X} \mid \boldsymbol{\theta}, \mathcal{H})$ that depend on $\rho(j)$ can be written in the form $\sum_j s(j) \log \rho(j)$, where $s(j)$ are the expected counts of how many times $\rho(j)$ was used. If we assume the counts $s(j)$ to be constant and vary only the probabilities $\rho(j)$, we can find the maximum of the a posteriori density analytically:

$$\rho(i) = \frac{s(i) + z(i) - 1}{\sum_j [s(j) + z(j) - 1]}. \tag{11}$$

The solution of the componentwise Bayes estimate [8]

$$\rho(i) = \frac{s(i) + z(i)}{\sum_j [s(j) + z(j)]} \tag{12}$$

resembles closely the map solution (11). The cB estimate is actually a map-estimate with a modified Dirichlet prior (the pseudocounts $z(j)$ increased by one), i.e. the same convergence results hold.

5 EXPERIMENTS

Two sets of experiments were made. The first one highlights the differences of the estimators on a very simple example. The second one shows how the parameters for the example given in (1) are learned from generated data.

5.1 Comparison of Estimators

Let us consider the following simple HMM written as a LOHMM:

$$\begin{array}{lll} c(1) \xleftarrow{0.5} start. & c(1) \xleftarrow{0.9:h} c(1). & c(2) \xleftarrow{0.1:h} c(2). \\ c(2) \xleftarrow{0.5} start. & c(1) \xleftarrow{0.1:t} c(1). & c(2) \xleftarrow{0.9:t} c(2). \end{array}$$

This could be interpreted as someone picking either coin 1 or coin 2 with equal probability and then using that coin to generate a sequence of heads and tails. Coin 1 produces more heads and coin 2 more tails.

Given sequences h, h, h and t, t, t as data, the structure shown above and a uniform prior over parameters, we now ask, what would the different estimators give as parameters. In fact, all of them would give 0.5 for the selection between the coins, but the probabilities p_1 and p_2 for the coins 1 and 2 to produce heads are of more interest.

There are three fixed points for the maximum a posteriori (or maximum likelihood) estimator (Eq. 3). The first one is a saddle point at $p_1 = 0.5$, $p_2 = 0.5$ and the two others are the global maxima $p_1 = 1.0$, $p_2 = 0.0$ and $p_1 = 0.0$, $p_2 = 1.0$. Using random initialisation, the Baum-Welch algorithm would end up in the latter two with equal probabilities and to the first one with probability 0. This estimator would conclude from the data that one of the coins produces heads every time and the other only tails. If the estimated model is tested with a sequence h, t, h, it would give a likelihood of exactly 0. From this failure one could conclude, that maximum likelihood estimator does not prepare well for new data if there is a limited amount of data available for learning.

The Bayes estimator (Eq. 4) is hard to evaluate in general, but in this case one can use symmetry to conclude that it is $p_1 = 0.5$, $p_2 = 0.5$. Since the estimator is always unique, it cannot decide which coin produces more heads (resp. tails).

The componentwise Bayes estimator (Eq. 5) has also three fixed points. The first one is the saddle point at $p_1 = 0.5$, $p_2 = 0.5$ in analogy with the map-estimator. The stable points are now at $p_1 = 0.789$, $p_2 = 0.211$ and $p_1 = 0.211$, $p_2 = 0.789$. Again, random initialisation will decide which one is chosen. The cB estimator seems to combine the good properties of the other two. It can operate with limited amount of data like the Bayes estimator, but can avoid the symmetrical solution.

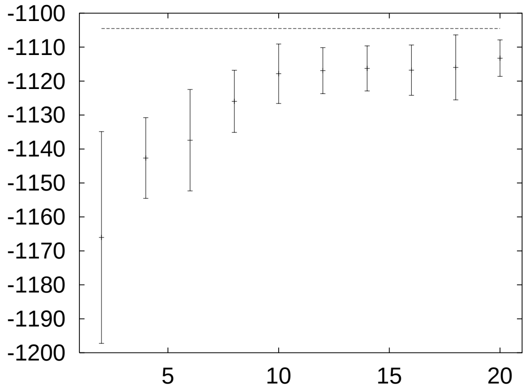

Figure 1: *Log likelihood of the testing set as the function of number of sequences used for training. The mean and the standard deviation of 5 runs is shown. The horizontal line corresponds to the LOHMM which generated the data.*

5.2 Re-learning parameters

We implemented the algorithms for LOHMM using Sicstus-3.8.6 Prolog. The LOHMM in (1) was used to generate data sequences of length 10. The train set consisted of 20 sequences and the test set consisted of 50 sequences. The parameters of the LOHMM were randomly initialised and estimated from fraction of the training data using the componentwise Bayes estimator. It took about 10 iterations to reach our stopping criterion: increase of less than 0.1 in the log likelihood of the training data. We measured the likelihood of the test set for each learned model. Figure 1 shows how the results (expectedly) get better with more data to learn from.

6 RELATED WORK

HMMs have been extended in a number of different ways e.g. hierarchical HMMs [3], factorial HMMs [6] and based on tree automata [4]. Relational Markov Models (RMMs) [1] also use logical representations, but they do not allow for variable binding, unification nor hidden states. LOHMMs are introduced in [10] concentrating on the application to the secondary structure of proteins.

7 CONCLUSIONS

In this paper we showed, how to learn LOHMMs from data. After comparison, we chose the componentwise Bayes estimator, which seems to best fit our needs. We ran experiments with synthetic data to show that the presented algorithms work.

In near future, we concentrate our research on the structural learning of LOHMMs and on different kinds of selection distributions μ.

8 ACKNOWLEDGEMENTS

T. Raiko was supported by a Marie Curie fellowship at DAISY, HPMT-CT-2001-00251.

References

[1] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, 2002.

[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[3] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: analysis and applications. *Machine Learning*, 32, 1998.

[4] P. Frasconi, G. Soda, and A. Vullo. Hidden Markov models for text categorization in multi-page documents. *Journal of Intelligent Information Systems, (special issue on Automated Text Categorization)*, 18:195–217, 2002.

[5] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, New York, 1995.

[6] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

[7] Geoffrey E. Hinton and Drew van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proc. COLT'93*, pages 5–13, Santa Cruz, California, USA, July 26–28, 1993.

[8] K. Karplus. Regularizers for estimating distributions of amino acids from small samples. Technical Report UCSC-CRL-95-11, University of California, Santa Cruz, 1995.

[9] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden Markov models. Technical Report 175, University of Freiburg, Germany, June 2002.

[10] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden Markov models. In *Proceedings of the Pacific Symposium on Biocomputing*, 2003. (to appear).

[11] J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 2. edition, 1989.

[12] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 1994.

[13] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), 1989.