Fisher Kernels for Logical Sequences

Kristian Kersting[1] and Thomas Gärtner[2]

[1] University of Freiburg, Institute for Computer Science, Machine Learning Lab, Georges-Koehler-Allee 079, 79110 Freiburg, Germany
kersting@informatik.uni-freiburg.de
[2] Fraunhofer Institut Autonome Intelligente Systeme, Knowledge Discovery Team, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
thomas.gaertner@ais.fraunhofer.de

Abstract. One approach to improve the accuracy of classifications based on generative models is to combine them with successful discriminative algorithms. *Fisher kernels* were developed to combine generative models with a currently very popular class of learning algorithms, kernel methods. Empirically, the combination of hidden Markov models with support vector machines has shown promising results. So far, however, Fisher kernels have only been considered for sequences over flat alphabets. This is mostly due to the lack of a method for computing the gradient of a generative model over structured sequences. In this paper, we show how to compute the gradient of *logical hidden Markov models*, which allow for the modelling of logical sequences, i.e., sequences over an alphabet of logical atoms. Experiments show a considerable improvement over results achieved without Fisher kernels for logical sequences.
Keywords: Hidden Markov Models, Fisher Kernels, Logical Sequences

1 Introduction

Generative models in general and hidden Markov models (HMMs) [17] in particular are widely used in computational biology. One of their application areas there is protein fold recognition where one tries to understand how proteins fold up in nature. Usually, for every protein with unknown structure one aims at finding the most similar protein with known structure (or fold) in a database.

From a machine learning perspective, fold recognition can be regarded as a classification problem: One tries to estimate the dependence of a target variable y on some observation x, based on a finite set of observations for which the value of the target variable is known. More technically, given a finite set of training examples $\{(x_i, y_i)\}_{i=1}^m \subseteq \mathcal{X} \times \{-1, +1\}$, one tries to find a function $f : \mathcal{X} \to \{-1, +1\}$ with low approximation error on the training data as well as on unseen examples. For HMMs it is common to use the *plug-in estimate*

$$f(x) = \mathrm{sign}[P(y = +1 \mid x, \theta^*) - 0.5]$$

The maximum likelihood parameters θ^* of the given HMM are usually estimated using the EM algorithm, for example by the Baum-Welch procedure. Despite of their success, HMMs have two major weaknesses:

(A) they are able to only handle sequences over flat alphabets, and
(B) the predictive performance of the plug-in estimate is often lower than that of discriminative classifiers.

To overcome **(A)**, *logical hidden Markov models* (LOHMMs) [14] have recently been introduced as an extension of HMMs. They allow for *logical sequences*, i.e., sequences of atoms in a first order logic. In [14], LOHMMs have been applied to the problem of discovering structural signatures of protein folds and led to more compact models. The trained LOHMM consisted of 120 parameters corresponding to an HMM with more than 62000 parameters.

To overcome **(B)**, i.e., to improve the classification accuracy of generative models — usually HMMs — different kernel functions have been proposed in order to make use of the good predictive performance of kernel methods such as the support vector machine (SVM) [19]. The most prominent of these kernel functions is the *Fisher kernel* [11]. The key idea there is to use the gradient of the log likelihood of the generative model with respect to its parameters as feature vector. The motivation to use this feature space is that the gradient of the log-likelihood with respect to the parameters of a generative model captures the generative process of a sequence better than just the posterior probabilities.

Fisher kernels have successfully been applied in many learning problems where the instances are sequences over a flat alphabet. Many sequences occurring in real-world problems, however, exhibit internal structure. The elements of such sequences can be seen as atoms in a first order logic (see e.g. [16] for an introduction to first order logic). For example, the secondary structure of the Ribosomal protein L4 can be represented as

$$\begin{aligned}&\mathtt{st}(\mathtt{null},2),\mathtt{he}(\mathtt{h}(\mathtt{right},\mathtt{alpha}),6),\mathtt{st}(\mathtt{plus},2),\mathtt{he}(\mathtt{h}(\mathtt{right},\mathtt{alpha}),4),\\&\mathtt{st}(\mathtt{plus},2),\mathtt{he}(\mathtt{h}(\mathtt{right},\mathtt{alpha}),4),\mathtt{st}(\mathtt{plus},3),\mathtt{he}(\mathtt{h}(\mathtt{right},\mathtt{alpha}),4),\\&\mathtt{st}(\mathtt{plus},1),\mathtt{he}(\mathtt{h}(\mathtt{right},\mathtt{alpha}),6)\end{aligned} \tag{1}$$

Here, helices of a certain type and length $\mathtt{he}(\mathit{HelixType},\mathit{Length})$, and strands of a certain orientation and length $\mathtt{st}(\mathit{Orientation},\mathit{Length})$ are structured symbols, i.e., atoms over logical predicates. It has been argued that using secondary structure information is likely to improve fold recognition results, see e.g. [7]. The application of HMMs to such sequences requires one to either ignore the structure of helices and strands, which results in a loss of information, or to take all possible combinations (of arguments such as orientation and length) into account, which leads to a combinatorial explosion in the number of parameters.

The main contribution of this paper is a method to compute the gradient of the log likelihood with respect to the parameters of a LOHMM. Though we focus on classification using Fisher kernels and SVMs, in general, such a method can also be used to devise fast gradient-based methods [18] and accelerated EM algorithms [4] for parameter estimation. We empirically compare the accuracy achieved in [14] with the accuracy achieved by Fisher kernels and SVMs using the same LOHMMs and datasets as in [14]. Furthermore, we conduct experiments on tree structured data in a mRNA signal structure detection task. Both results show that the predictive accuracy of LOHMMs can considerably be improved using Fisher kernels and SVMs.

The outline of the paper is as follows. In Sections 2 and 3 we briefly review Fisher kernels and LOHMMs. Then, in Section 4, we devise a method to compute the gradient of the log likelihood of LOHMMs which is essential to define Fisher kernels for logical sequences. In Section 5, we experimentally evaluate Fisher kernels for logical sequence. Before concluding, we discuss related work.

2 Kernels Methods and Probabilistic Models

Support vector machines [19] are a kernel method that can be applied to binary supervised classification problems. Being on one hand theoretically well founded in statistical learning theory, they have on the other hand shown good empirical results in many applications.

The characteristic aspect of this class of learning algorithms is the formation of hypotheses by linear combination of positive-definite kernel functions 'centred' at individual training examples. It is known that such functions can be interpret as the inner product in a Hilbert Space. The solution of the support vector machine is then the hyperplane in this Hilbert space that separates positive and negative labelled examples, and is at the same time maximally distant from the convex hulls of the positive and the negative examples. Conversely, every inner product in a linear space is a positive-definite kernel function.

Approaches for defining valid kernel functions for a given type of data can be distinguished as *syntax-driven* or *model-driven* [5]. We summarise some syntax-driven kernels in Section 6. The most prominent model-driven approach is the *Fisher kernel*. There, a kernel function is derived from a generative probability model of the domain. More precisely, every learning example is mapped to the gradient of the log likelihood of the generative model with respect to its parameters. The kernel is then the inner product of the examples' images under this map. The motivation to use this feature space is that the gradient of the log-likelihood with respect to the parameters of a generative model captures the generative process of a sequence better than just the posterior probabilities.

Given a parametric probability model M with parameters $\theta = (\theta_1, \ldots, \theta_n)^\top$, maximum likelihood parameters θ^*, and output probability $P(x \mid \theta, M)$, the Fisher score mapping U_x is defined as

$$U_x = \nabla_\theta \log P(x \mid \theta^*, M) = \left(\frac{\partial \log P(x \mid \theta^*, M)}{\partial \theta_1}, \ldots, \frac{\partial \log P(x \mid \theta^*, M)}{\partial \theta_n} \right)^\top .$$

The Fisher information matrix is the expectation of the outer product of the Fisher scores over $P(x \mid \theta, M)$, more precisely,

$$J_\theta = E_x \left[\nabla_\theta \log P(x \mid \theta, M)\right] \left[\nabla_\theta \log P(x \mid \theta, M)\right]^\top .$$

Given these definitions, the Fisher kernel is defined as

$$k(x, x') = U_x^\top J_{\theta^*}^{-1} U_{x'} = \left[\nabla_\theta \log P(x \mid \theta^*, M)\right]^\top J_{\theta^*}^{-1} \left[\nabla_\theta \log P(x' \mid \theta^*, M)\right] . \quad (2)$$

In practice often the role of the Fisher information matrix J_θ is ignored, yielding the kernel $k(x, x') = U_x^\top U_{x'}$. In the remainder of this paper we will follow this habit mainly to reduce the complexity of computation.

3 Probabilistic Models for Logical Sequences

The logical component of HMMs corresponds to a *Mealy machine*, i.e., to a finite state machine where output symbols are associated with transitions. This is essentially a propositional representation because the symbols used to represent states and outputs are flat, i.e., not structured. The key idea to develop a probabilistic model for structured sequences is to replace these flat symbols by abstract symbols. *Logical hidden Markov models* (LOHMMs) replace them by logical atoms. In this section, we will briefly review LOHMMs [15, 13].

First-Order Predicate Logic: Based on the representation of the Ribsomal protein $L4$ given in (1) we describe the necessary concepts of first-oder predicate logic. The symbols `st`, `null`, `2`, `he`, `h`, ... are distinguished into predicate and function symbols. Associated with every symbol is the *arity*, i.e., number of arguments. In the example, `st/2` and `he/2` are predicates of arity 2, `h/2` is a function of arity 2, and `plus/0`, `1/0`, ... are functions of arity 0, i.e., constants. The *first-order logic alphabet* Σ consists of predicates, functions, and variables (e.g., `X`). A *term* is a variable or a function symbol followed by its arguments in brackets such as $\mathtt{h}(\mathtt{right}, \mathtt{X})$ or `4`; an *atom* is a predicate symbol followed by its arguments in brackets such as $\mathtt{he}(\mathtt{h}(\mathtt{right}, \mathtt{X}), 4)$. Valid arguments of functions and predicates are terms. That one atom is a logical consequence of another is denoted by an *iterative clause*, such as $\mathtt{st}(\mathtt{X}, 12) \leftarrow \mathtt{st}(\mathtt{X}, 10)$. A *ground term*, *atom*, or *clause* is one that does not contain any variables. In the protein example $\mathtt{st}(\mathtt{null}, 2)$, $\mathtt{he}(\mathtt{h}(\mathtt{right}, \mathtt{alpha}), 6)$, ... are ground atoms and `null`, `2`, $\mathtt{h}(\mathtt{right}, \mathtt{alpha})$, `right`, `alpha`, ... are ground terms. A substitution $\sigma = \{\mathtt{X}/\mathtt{plus}\}$ is an assignment of terms `plus` to variables `X`. Applying a substitution σ to a term, atom or clause e yields the instantiated term, atom, or clause $e\sigma$ where all occurrences of the variables `X` are simultaneously replaced by the term `plus`, e.g., $(\mathtt{st}(\mathtt{X}, 12) \leftarrow \mathtt{st}(\mathtt{X}, 10))\{\mathtt{X}/\mathtt{plus}\}$ yields $\mathtt{st}(\mathtt{plus}, 12) \leftarrow \mathtt{st}(\mathtt{plus}, 10)$. The set of all ground atoms constructed from an alphabet Σ is the Herbrand base hb_Σ. For every atom A we define $G_\Sigma(A) = \{A\sigma \in \mathrm{hb}_\Sigma : \text{substitution } \sigma\}$. A substitution σ is a *unifier* of a finite set of atoms S if $S\sigma$ is singleton; if furthermore for every unifier σ' of S there is a substitution σ'' such that $\sigma = \sigma'\sigma''$ then σ is the *most general unifier* (MGU) of S.

Logical Hidden Markov Models (LOHMMs): The sequences generated by LOHMMs are sequences of ground atoms rather than flat symbols. Within LOHMMs, the flat symbols employed in traditional HMMs are replaced by logical atoms such as $\mathtt{st}(\mathtt{X}, 10)$. Each atom $\mathtt{st}(\mathtt{X}, 10)$ there represents the set of ground atoms $G_\Sigma(\mathtt{st}(\mathtt{X}, 10))$.

Additionally, we assume that the alphabet is typed which in our case means that there is a function mapping every predicate `r/m` and number $1 \leq i \leq m$ to

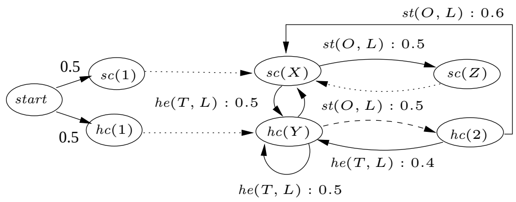

Fig. 1. A logical hidden Markov model. The vertices denote abstract (hidden) states where hc(ID) (resp. sc(ID)) represents a block ID of consecutive helices (resp. strands). Solid edges encode abstract transitions. Dotted edges indicate that two abstract states behave in exactly the same way. Dashed edges represent a *more general than* relation.

the set of ground terms allowed as the i-th argument of predicate r/m. This set is called the domain of the i-th argument of predicate r/m.

Figure 1 shows a LOHMM graphically. The states, observations, and transitions of LOHMMs are **abstract** in the sense that every abstract state or observation A represents all possible concrete states in $G_\Sigma(\mathtt{A})$. In Figure 1 *solid edges* encode **abstract transitions**. Let H and B be logical atoms representing abstract states, let O be a logical atom representing an abstract output symbol. An abstract transition from state B with probability p to state H and omitting O is denoted by $p : \mathtt{H} \xleftarrow{\mathtt{O}} \mathtt{B}$. If H, B, and O are all ground, there is no difference to 'normal' transitions. Otherwise, if H, B, and O have no variables in common, the only difference to 'normal' transitions is that for each abstract state (resp. observation) we have to sample which concrete state (resp. observation) we are in. Otherwise, we have to remember the variable bindings. More formally, let $\mathtt{B}\sigma_\mathtt{B} \in G_\Sigma(\mathtt{B})$, $\mathtt{H}\sigma_\mathtt{B}\sigma_\mathtt{H} \in G_\Sigma(\mathtt{H}\sigma_\mathtt{B})$, $\mathtt{O}\sigma_\mathtt{B}\sigma_\mathtt{H}\sigma_\mathtt{O} \in G_\Sigma(\mathtt{O}\sigma_\mathtt{B}\sigma_\mathtt{H})$, and let μ be a **selection distribution**. Then with probability $p \cdot \mu(\mathtt{H}\sigma_\mathtt{B}\sigma_\mathtt{H} \mid \mathtt{H}\sigma_\mathtt{B}) \cdot \mu(\mathtt{O}\sigma_\mathtt{B}\sigma_\mathtt{H}\sigma_\mathtt{O} \mid \mathtt{O}\sigma_\mathtt{B}\sigma_\mathtt{H})$, the model makes a transition from state $\mathtt{B}\sigma_\mathtt{B}$ to $\mathtt{H}\sigma_\mathtt{B}\sigma_\mathtt{H}$ and emits symbol $\mathtt{O}\sigma_\mathtt{B}\sigma_\mathtt{H}\sigma_\mathtt{O}$.

A selection distribution specifies for each abstract state (respectively observation) A over the alphabet Σ a distribution $\mu(\cdot \mid \mathtt{A})$ over $G_\Sigma(\mathtt{A})$.Consider, for example, the abstract transition $0.5 : \mathtt{s(Z)} \xleftarrow{\mathtt{o(X,Y,Z)}} \mathtt{s(X)}$. Suppose, $\mathtt{B}\sigma_\mathtt{B} = \mathtt{s(1)}$, $\mu(\mathtt{s(3)} \mid \mathtt{s(Z)}) = 0.2$, and $\mu(\mathtt{o(1,2,3)} \mid \mathtt{o(1,Y,3)}) = 0.05$. Then, from state s(1) with probability $0.5 \times 0.2 \times 0.05 = 0.005$ the output symbol is o(1,2,3) and the next state is s(3). To reduce the model complexity, we employ a naïve Bayes approach. For each domain D_i there is a probability distribution P_{D_i}. Let vars(A) $= \{\mathtt{V_1}, \ldots, \mathtt{V_l}\}$ be the variables occurring in A, and let $\sigma = \{\mathtt{s_1}/\mathtt{V_1}, \ldots \mathtt{s_l}/\mathtt{V_l}\}$ be a substitution grounding A. Each $\mathtt{V_j}$ is then considered a random variable over the domain of the first argument of r/m it appears in, denoted by $D_{\mathtt{V_j}}$. Then, $\mu(\mathtt{A}\sigma \mid \mathtt{A}) = \prod_{j=1}^{l} P_{D_{\mathtt{V_j}}}(\mathtt{V_j} = \mathtt{s_j})$.

Indeed, multiple abstract transitions can match a given ground state. Consider hc(Y) and hc(2) in Figure 1. For the state hc(2) the matching abstract transitions do not sum to 1.0. To resolve this, we only consider the maximally specific transitions (with respect to the body parts B) that apply to a state in order to determine the successor states. This **conflict resolution strategy** is encoded in Figure 1 by *dashed edges* which represent the *more-general-than* relation among abstract states. It ensures that for $\mathtt{hc(2)}$ only the outgoing transitions of $\mathtt{hc(2)}$ fire whereas for $\mathtt{hc(3)}$ only the outgoing transitions of $\mathtt{hc(Y)}$ fire. The rational behind this is that if there exists a substitution σ such that $\mathtt{B}_2\sigma = \mathtt{B}_1$, i.e., $\mathtt{B}_2$ subsumes $\mathtt{B}_1$, then the first transition can therefore be regarded as more informative than the second one because $G_\Sigma(\mathtt{B}_1) \subseteq G_\Sigma(\mathtt{B}_2)$.

Finally, *dotted edges* indicate that two abstract states behave in exactly the same way. If we follow a transition to an abstract state with an outgoing dotted edge, we will automatically follow that edge making appropriate unifications.

Definition 1 *A* logical hidden Markov model *(LOHMM) is a tuple* $M = (\Sigma, \mu, \Delta)$ *where* Σ *is a first-order logical alphabet,* μ *a selection probability over* Σ*, and* Δ *is a set of abstract transitions. Let* $\mathbf{B}$ *be the set of all atoms that occur as the body part of transitions in* Δ*. We require* $\forall \mathtt{B} \in \mathbf{B} : \sum_{p:\mathtt{H} \xleftarrow{\mathtt{O}} \mathtt{B} \in \Delta} p = 1$.

In [13] it is proven that LOHMMs specify a unique probability measure over hb_Σ. Here, we would like to exemplify that LOHMMs are generative models. Consider the model in Figure 1. Starting from $\mathtt{start}$, it chooses an initial abstract state, say $\mathtt{hc(1)}$. Forced to follow the dotted edge, it enters the abstract state $\mathtt{hc(Y)}$. In each abstract state, the model samples values for all variables that are not instantiated yet according to the *selection distribution* μ. Since the value of $\mathtt{Y}$ was already instantiated in the previous abstract state $\mathtt{hc(1)}$, it does not sample a value for $\mathtt{Y}$. Now, it selects a transition, say to $\mathtt{hc(Y)}$, observing $\mathtt{he(T,L)}$. Since $\mathtt{Y}$ is shared among the head and the body, the state $\mathtt{hc(1)}$ is selected with probability 1.0. The observation $\mathtt{he(h(right,3to10),10)}$ is sampled from $\mathtt{he(T,L)}$ using μ. Now, the model goes over to $\mathtt{sc(X)}$, emitting $\mathtt{st(plus,10)}$ which in turn was sampled from $\mathtt{st(O,L)}$. Variable $\mathtt{X}$ in $\mathtt{sc(X)}$ is not yet bound; so, a value, say $\mathtt{2}$, is sampled using μ. Next, we move on to abstract state $\mathtt{sc(Z)}$, emitting $\mathtt{st(plus,15)}$. The variable $\mathtt{Z}$ is sampled to be $\mathtt{3}$. The dotted edge brings us back to $\mathtt{sc(X)}$ and automatically unifies $\mathtt{X}$ with $\mathtt{Z}$, which is bound to $\mathtt{3}$. Emitting $\mathtt{he(h(right,alpha),9)}$, the model returns to abstract state $\mathtt{hc(Y)}$. Assume that it samples $\mathtt{2}$ for variable $\mathtt{Y}$, it has to follow the dashed outgoing edge to $\mathtt{hc(2)}$.

4 Fisher Kernels for Logical Sequences

In Equation (2), we gave the definition of the Fisher kernel based on the gradient of the log likelihood of a set of observations with respect to the parameters θ of a generative Model M. In this section we derive a Fisher Kernel for logical sequences by employing LOHMMs. Let $\mathtt{O} = \{\mathtt{O}_1, \ldots, \mathtt{O}_m\}$ be a set of ground observation sequences. A single ground observation sequence $\mathtt{O}_i$ consists of a sequence $\mathtt{o}_{i1}, \mathtt{o}_{i2}, \ldots, \mathtt{o}_{iT_i}$ of ground atoms. We assume that $T_1 = T_2 = \ldots = T_m$ and that the $\mathtt{O}_i$ are independently identically distributed (iid). Thus,

$$\frac{\partial \log P(\mathtt{O} \mid \theta, M)}{\partial \theta} = \sum_{k=1}^{m} \frac{\partial \log P(\mathtt{O}_k \mid \theta, M)}{\partial \theta} = \sum_{k=1}^{m} \frac{1}{P(\mathtt{O}_k \mid \theta, M)} \cdot \underbrace{\frac{\partial P(\mathtt{O}_k \mid \theta, M)}{\partial \theta}}_{(*)}$$

The key step to derive the gradient formula is to rewrite:

$$
\begin{aligned}
P(\mathtt{O}_k \mid \theta, M) &= \sum_{\mathtt{s} \in \mathbf{S}_t} P(\mathtt{o_{k1}}, \ldots, \mathtt{o_{kt}}, s_t = \mathtt{s} \mid \theta, M) \cdot P(\mathtt{o_{kt+1}}, \ldots, \mathtt{o_{kT_k}} \mid s_t = \mathtt{s}, \theta, M) \\
&= \sum_{\mathtt{s} \in \mathbf{S}_t} \alpha_t(\mathtt{s}) \cdot \beta_t(\mathtt{s}) \qquad (3)
\end{aligned}
$$

where $\alpha_t(\mathtt{s})$ is the forward probability of state $\mathtt{s}$ and $\beta_t(\mathtt{s})$ is the backward probability of state $\mathtt{s}$ for $\mathtt{O}_k$. The term S_t denotes the set of hidden states the system can be in at time t.

The parameter vector θ defines the set of parameters for all abstract transitions and for all selection distributions in the LOHMM. We will now show how to compute the partial derivatives $(*)$ for transition probabilities and for selection probabilities μ separately.

Abstract Transitions: Let θ_{ij} be an abstract transition probability, i.e., a probability value associated to the jth abstract transition $\mathtt{T} \equiv \theta_{ij} : \mathtt{H} \xleftarrow{\mathtt{O}} \mathtt{B}$ of the ith abstract body $\mathtt{B}$ in $\mathbf{B}$. Due to the chain rule it holds

$$
\frac{\partial P(\mathtt{O}_k \mid \theta, M)}{\partial \theta_{ij}} = \sum_{t=0}^{T+1} \sum_{\mathtt{s_H} \in S_t} \frac{\partial P(\mathtt{O}_k \mid \theta, M)}{\partial \alpha_t(\mathtt{s_H})} \times \frac{\partial \alpha_t(\mathtt{s_H})}{\partial \theta_{ij}}. \qquad (4)
$$

By independence of $\alpha_t(\mathtt{s_H})$ and $\beta_t(\mathtt{s_H})$ in Equation (3):

$$
\frac{\partial P(\mathtt{O}_k \mid \theta, M)}{\partial \alpha_t(\mathtt{s_H})} = \beta_t(\mathtt{s_H}) \qquad (5)
$$

The partial derivative of $\alpha_t(\mathtt{s_H})$ w.r.t. θ_{ij} in Equation (4) can then be deduced from the *forward procedure*, see e.g. [15]:

$$
\frac{\partial \alpha_t(\mathtt{s_H})}{\partial \theta_{ij}} = \sum_{\mathtt{s_B} \in S_{t-1}} \xi(\mathtt{T}, \mathtt{s_B}, \mathtt{s_H}, \mathtt{o}_{kt-1}) \cdot \alpha_{t-1}(\mathtt{s_B}) \cdot \mu(\mathtt{s_H} \mid \mathtt{H}\sigma_{\mathtt{s_B}}) \cdot \mu(\mathtt{o}_{kt-1} \mid \mathtt{O}\sigma_{\mathtt{s_B}}\sigma_{\mathtt{s_H}}),
$$

where $\xi(\mathtt{T}, \mathtt{s_B}, \mathtt{s_H}, \mathtt{o}_{kt-1})$ indicates that 1) $\mathtt{B}$ is maximally specific for $\mathtt{s_B}$, 2) $\mathtt{s_H}$ unifies with $\mathtt{H}$, and 3) $\mathtt{o}_{kt-1}$ unifies with $\mathtt{O}$, and σ_{-} are the corresponding MGUs.

Selection Distribution: Now, let θ_{ij} be a selection probability value. Let $\mathtt{r/n}$ be a predicate with domains $D_1, \ldots, D_n$, where $D_i = \{d_{i1}, \ldots, d_{im_i}\}$. Furthermore, assume that the the selection distribution for $\mathtt{r}$ is specified by $\theta_{ij} = P(D_i = d_{ij})$. Equations (4) and (5) remain the same. The term $\partial \alpha_t(\mathtt{s})/\partial \theta_{ij}$ is zero whenever d_{ij} was not *selected* to "ground" $\mathtt{H}\sigma_{\mathtt{b}}$ or $\mathtt{O}\sigma_{\mathtt{b}}\sigma_{\mathtt{h}}$. Because, the selection distribution follows a naïve Bayes scheme and $\frac{\partial x^m}{\partial x} = m \cdot x^{m-1} = m \cdot \frac{x^m}{x}$, this yields:

$$
\begin{aligned}
\frac{\partial \alpha_t(\mathtt{s_H})}{\partial \theta_{ij}} = \textstyle\sum_{\mathtt{s_B} \in S_{t-1}} \sum_{\mathtt{T} \equiv p : \mathtt{H} \xleftarrow{\mathtt{O}} \mathtt{B} \in \Delta} & \xi(\mathtt{T}, \mathtt{s_B}, \mathtt{s_H}, \mathtt{o}_{kt-1}) \cdot c_{ij}(\mathtt{s_B}, \mathtt{s_H}, \mathtt{o}_{kt-1}) \cdot \\
& \cdot \frac{\alpha_{t-1}(\mathtt{s_B}) \cdot p \cdot \mu(\mathtt{s_H} \mid \mathtt{H}\sigma_{\mathtt{s_B}}) \cdot \mu(\mathtt{o}_{kt-1} \mid \mathtt{O}\sigma_{\mathtt{s_B}}\sigma_{\mathtt{s_H}})}{\theta_{ij}}, \qquad (6)
\end{aligned}
$$

where $c_{ij}(\mathtt{T}, \mathtt{s_B}, \mathtt{s}, \mathtt{o}_{kt-1})$ denotes the number of times, the domain element d_{ij} has been selected in order to ground $\mathtt{s_B}$, $\mathtt{s}$, and $\mathtt{o}_{kt-1}$ when following abstract transition $\mathtt{T}$.

Constraint Satisfaction: So far, we have not taken the constraint into account that the parameter vector consists of probability values, i.e. $\theta_{ij} \in [0,1]$ and $\sum_j \theta_{ij} = 1$. A general solution, which we used in our experiments, is to reparameterise the problem so that the new parameters automatically respect the constraints on θ_{ij} no matter what their values are. To do so, we define the parameters $\bar{\theta}_{ij} \in \mathbb{R}$ such that $\theta_{ij} = exp(\bar{\theta}_{ij})/(\sum_l exp(\bar{\theta}_{il}))$. This enforces the constraints given above, and a local maximum w.r.t. $\bar{\theta}$ is also a local maximum w.r.t. θ, and vice versa. The gradient w.r.t the $\bar{\theta}$ can be found by computing the gradient w.r.t the θ and then deriving the gradient w.r.t. $\bar{\theta}$ using the chain rule.

5 Experiments

Having described how to compute the gradient of the log likelihood of LOHMMs with respect to its parameters, we are now ready to experimentally evaluate Fisher Kernels of LOHMMs. In this section we thus compare results achieved by LOHMMs alone with results achieved by LOHMMs combined with Fisher kernels. The experiments put the following hypothesis to test:

H The predictive accuracy of LOHMMs can be improved considerably using Fisher kernels and SVMs.

The experiments took place in two different bioinformatical domains: Protein fold recognition and mRNA signal structure detection. Both problems are multiclass problems with 5 different classes each. In order to tackle the multiclass problem with SVMs, we create for each class a binary classification problem, treating instances of this class as positive and all other instances as negative (one-against-all). As all binary classification problems consist of the same instances and the SVMs on each classification problem were trained with the same parameters, the resulting models are comparable. That is, to create a multiclass classification we compare the numerical output of the binary support vector machines on each test instance, and assign the class corresponding to the maximal numerical output. Finally, to overcome the problem that the number of instances per class strongly varies between classes, we set the misclassification cost in each binary problem to the fraction of positive instances in that dataset. The SVM implementation used in our experiments was SVM-light [12].

mRNA Signal Structure Detection: This experiment is concerned with identifying subsequences in mRNA that are responsible for biological functions [3]. In contrast to the secondary structure of proteins that form chains (see next experiment), the secondary structure of mRNAs form trees. As trees can not easily

[3] The *Science Magazine* listed RNA as one of the runner-up breakthroughs of the year 2003.

be handled using HMMs, mRNA secondary structure data is more interesting than that of proteins.

The first application of machine learning to recognise the signal structure class of mRNA molecules was described in [9]. The dataset we used [4] is the one used in [13], where LOHMMs were applied with the plug-in estimate. In total, there are 93 logical sequences (in total 3122 ground atoms) composed of 15 and 5 SECIS (Selenocysteine Insertion Sequence), 27 IRE (Iron Responsive Element), 36 TAR (Trans Activating Region) and 10 histone stemloops.

As the dataset is rather small, we used leave-one-out error estimation and did not further optimise the SVM parameters. That is, we used a linear kernel and let SVM-light choose the default complexity constant. The error rate of LOHMMs with the plugin estimate of 4.3% could be reduced to 2.2% by using Fisher kernels. More precisely, the Fisher kernels managed to resolve two misclassifications, one of IRE and one of SECIS. The result improves the error rate of 4.6% reported in [9]. This suggests that hypothesis **H** holds.

Protein Fold Recognition: This experiment is concerned with how proteins fold up in nature. This is an important problem, as the biological functions of proteins depend on the way they fold up. A common approach to protein fold recognition is to start from a protein with unknown structure and search for the most similar protein with known structure (or protein fold) in the database. This approach has been followed in [14] where LOHMMs with the plugin estimate were able to considerably improve over these results. Notice that the number of parameters of the LOHMMs used were by an order of magnitude smaller than the number of an equivalent HMM (120 vs. approx. 62000).

The data consists of logical sequences of the secondary structure of protein domains [5]. The task is to predict one of five SCOP [10] folds for 2187 test sequences given a LOHMM trained on 200 training sequences per fold. As this dataset is bigger than the previous, we were able to perform a proper parameter selection. We first performed a leave-one-out error estimation in the training set to choose the parameter of the Gaussian kernel function. Of the tested parameters ($\gamma \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$), $\gamma = 10^{-3}$ clearly performed best over all binary problems. We then fixed this parameter and optimised the complexity constant. Of the tested parameters ($C \in \{10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$), $C = 100$ clearly performed best over all binary problems (testing bigger values was not necessary, as we already achieved 0 unbounded support vectors).

Using Fisher kernels with the same LOHMMs as in [14] and the above described SVM parameters, we were able to reduce the error rate of the plugin

[4] The dataset is not the same as described in [9] due to problems in obtaining the original dataset. We will compare to the smaller data set used in [9] which consisted of 66 signal structures and is very close to our data set. On a larger data set (with 400 structures) [9] report an error rate of 3.8% .

[5] A domain can be viewed as a sub-section of a protein which appears in a number of distantly related proteins and which can fold independently of the rest of the protein.

	fold1	fold2	fold23	fold37	fold55
LOHMMs	0.86/0.78	0.69/0.67	0.56/0.71	0.72/0.66	0.86/0.96
Fisher Kernels	0.87/0.90	0.80/0.86	0.84/0.77	0.71/0.71	0.88/0.74

Table 1. Precision and recall values (precision/recall) for the protein fold experiment. The first row shows the values for LOHMMs alone. The second row shows the values for Fisher kernels of LOHMMs and SVMs.

estimate of 26% to an error rate of 17.4%. As Table 1 shows, the precision and recall values were well balanced within each class. This suggest that **H** holds.

6 Related Work

The past few years have witnessed a significant interest in *kernels* and *probabilistic models for structured data*. Despite this interest in both fields, to the best of our knowledge, the work of Taskar et al. [20] is the only work which aims at the same goal, namely discriminative models for structured data. However, Taskar et al. do not consider sequential but relational data and they do not explore kernel functions but discriminative learning of relational probabilistic models. Discriminative learning here means that instead of maximising the joint probability $P(y_i, x_i)$, the conditional probability $P(y_i \mid x_i)$ is maximised.

Support Vector Machines for Structured Data: In principle, there are two ways to apply support vector machines to structured data: Using *syntax-driven* and *model-driven* kernel functions. For an overview, we refer to [5].

An integral part of many *syntax-driven* kernels for structured data is the *decomposition* of an object into a set of its parts and the *intersection* of two sets of parts. The kernel on two objects is then defined as a measure of the intersection of the two corresponding sets of parts. In the case that the sets are finite or countable sets of vectors it is often beneficial to sum over all pairwise kernels on the elements. This idea of intersection and crossproduct kernels is reflected in most work on kernels for structured data, from the early and influential technical reports [8, 23] through work on string kernels, kernels for higher order terms and trees, to more recent work on graph kernels.

An alternative to syntax-driven kernel functions are *model-driven kernel* functions like the Fisher kernel introduced above. Based on the idea of maximising the the posterior probability estimated by the optimal logistic regressor in the extracted feature space, [21] introduced the so-called TOP kernel function. The TOP kernel function is the scalar product between the posterior log-odds of the model and the gradient thereof. The posterior log-odds is defined as the difference in the logarithm of the probability of each class given the instance. Marginalised kernels [22] have later been introduces as a generalisation of Fisher kernels. Here, a kernel over both the hidden and the observed data is assumed to be given. Then, the marginalised kernel for the visible data is obtained by taking the expectation with respect to hidden variables.

Probabilistic Models for Structured Data: LOHMMs combine two different research directions. On the one hand, they are related to several extensions of HMMs, such as hierarchical HMMs [3] or factorial HMMs [6]. Here, the underlying idea is to decompose the state variables into smaller units. For instance, to derive factorial HMMs, one factors the hidden state variable into k state variables which depend on one another only through the observation The key difference with LOHMMs is that these approaches do not employ the logical concept of unification. Unification is essential because it allows us to introduce abstract transitions, which do not consist of more detailed states.

On the other hand, they are also related to the recent interest in combining inductive logic programming principles with probability theory, see [2] for an overview. Most attention has been devoted to developing highly expressive formalisms. LOHMMs can be seen as an attempt towards *downgrading* such highly expressive frameworks. As a consequence, LOHMMs represent an interesting position on the expressiveness scale. Whereas they retain most of the essential logical features of the more expressive formalisms, they seem easier to understand, adapt and learn.

Most closely related to LOHMMs are *relational Markov models* [1]. Here, states can be of different types, with each type described by a different set of variables. The domain of each variable is hierarchically structured. The main difference is that variable bindings, unification, and hidden states are not used.

7 Conclusions

So far, Fisher kernels have only been considered for sequences of flat symbols. In this paper, Fisher kernels for logical sequences, i.e., sequences over an alphabet of logical atoms have been introduced and experimentally investigated. The experimental results show that Fisher kernels can handle logical sequences and that they can improve considerably the predictive performance of plug-in estimates of probabilistic models for logical sequences.

We are confident that Fisher kernels can be used to also improve the discriminative power of other approaches combining inductive logic programming principles with probability theory. Exploring this family of *logical Fisher kernels* is a promising future research direction.

Acknowledgements: The research was supported in part by the European Union under contract number FP6-508861, *Application of Probabilistic Inductive Logic Programming II*, and in part by the DFG project (WR 40/2-1) *Hybride Methoden und Systemarchitekturen für heterogene Informationsräume.*

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.

2. L. De Raedt and K. Kersting. Probabilistic Logic Learning. *ACM-SIGKDD Explorations*, 5(1):31–48, 2003.
3. S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: analysis and applications. *Machine Learning*, 32:41–62, 1998.
4. J. Fischer and K. Kersting. Scaled CGEM: A Fast Accelerated EM. In N. Lavrac, D. Gamberger, H. BLockeel, and L. Todorovski, editors, *Proceedings of the Fourteenth European Conference on Machine Learning (ECML-03)*, pages 133–144, Cavtat, Croatia, September 22–26 2003.
5. T. Gärtner. Kernel-based Learning in Multi-Relational Data Mining. *ACM-SIGKDD Explorations*, 5(1):49–58, 2003.
6. Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.
7. J. Hargbo and A. Elofsson. Hidden markov models that use predictied secondary structure for fold recognition. *Proteins: Structure, Function, and Genetics*, 36:68–76, 1999.
8. D. Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
9. T. Horváth, S. Wrobel, and U. Bohnebeck. Relational Instance-Based learning with Lists and Terms. *Machine Learning*, 43(1/2):53–80, 2001.
10. T. Hubbard, A. Murzin, S. Brenner, and C. Chotia. *SCOP*: a structural classification of proteins database. *NAR*, 27(1):236–239, 1997.
11. T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Adv. in Neural Information Processing Systems 11*. MIT Press, 1999.
12. T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer, 2002.
13. K. Kersting, L. De Raedt, and T. Raiko. Logical hidden markov models. 2004. (submitted).
14. 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 (PSB-03)*, 2003.
15. K. Kersting, T. Raiko, and L. De Raedt. A Structural GEM for Learning Logical Hidden Markov Models. In *Working Notes of the Second KDD-Workshop on Multi-Relational Data Mining (MRDM-03)*, 2003.
16. J. W. Lloyd. *Foundations of Logic Programming*. Springer, 2. edition, 1989.
17. L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
18. R. Salakhutdimov, S. Roweis, and Z. Ghahramani. Optimization with EM and Expectation-Conjugate-Gradient. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML03)*, 2003.
19. B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002.
20. B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 2002.
21. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. In *Adv. in Neural Information Processing Systems*. MIT Press, 2002.
22. K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 2002.
23. C. Watkins. Kernels from matching operations. Technical report, Department of Computer Science, Royal Holloway, University of London, 1999.