Fisher Kernels for Relational Data

Uwe Dick and Kristian Kersting

University of Freiburg, Institute for Computer Science, Machine Learning Lab,
Georges-Koehler-Allee, Building 079, 79110 Freiburg, Germany
{dick,kersting}@informatik.uni-freiburg.de

Abstract. Combining statistical and relational learning receives currently a lot of attention. The majority of statistical relational learning approaches focus on density estimation. For classification, however, it is well-known that the performance of such generative models is often lower than that of discriminative classifiers. One approach to improve the performance of generative models is to combine them with discriminative algorithms. Fisher kernels were developed to combine them with kernel methods, and have shown promising results for the combinations of support vector machines with (logical) hidden Markov models and Bayesian networks. So far, however, Fisher kernels have not been considered for relational data, i.e., data consisting of a collection of objects and relational among these objects. In this paper, we develop Fisher kernels for relational data and empirically show that they can significantly improve over the results achieved without Fisher kernels.

1 Introduction

From a machine learning perspective, many real world applications can be regarded as classification problems: One tries to estimate the dependence of a target variable Y on some observation $\mathbf{X}$, based on a finite set of observations $\mathbf{x}$ for which the value y of the target variable is known. More formally:

Definition 1 (Classification Problem). Given *a finite set of training examples* $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m} \subseteq \mathcal{X} \times \mathcal{Y}$ *, where* $\mathcal{X}$ *is the feature space and* $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ *is the set of possible classes,* **find** *a function* $f : \mathcal{X} \rightarrow \mathcal{Y}$ *with low approximation error on the training data as well as on unseen examples.*

The classification problem has traditionally been considered for data in attribute-value form, i.e., $\mathbf{x}_i$ is a vector of fixed length. Many — if not most — real-world data sets, however, are not in attribute-value form. Applications are characterized by the presence of uncertainty and complex relations. Consider the KDD Cup 2001 localization of genes/proteins task [1].

Example 1 (Genes/Proteins Localization). The KDD Cup 2001 focused on data from life science. One data set, which we also used in our experiments, is from genomics. The data consists of 1243 genes of one particular but unknown type of organism. Each gene encodes a protein, which occupies a particular position in some part of a cell. For each gene, information on the class, the phenotype, i.e., its characteristics, the complex it belongs to etc. are given. Furthermore, the graph of interactions among the genes is provided. Using relational logic, this can elegantly be represented as set of ground atoms $\texttt{gene(g1)}.$ $\texttt{gene(g2)}.$ $\ldots$ $\texttt{phenotype(g1,1)}.$ $\ldots$ $\texttt{complex(g2,13)}.$ $\ldots$ $\texttt{interaction(g1,g2)}.$ $\texttt{interaction(g3,g245)}$ $\ldots$ Here, $\texttt{gene}/1$, $\texttt{phenotyp}/2$, $\texttt{gene}/1$ $\texttt{interaction}/2$ are *predicates* that identify relations, numbers and lower-case strings like $\texttt{g1}$ and $\texttt{1}$ are *constants* that identify objects. *Ground atoms* are predicates together with their arguments, for example $\texttt{interaction(g3,g245)}$ denotes that genes $\texttt{g3}$ and $\texttt{g245}$ interact. 381 of the 1243 genes are withheld as test set. The task is to predict the localization of a protein/gene based on the features of the protein/gene and of proteins/genes interacting with the protein/gene and is characterized by the presence of uncertainty, a varying number of objects (genes), and relations (interactions) among the objects.

Inductive logic programming [17] (ILP) and relational learning have been developed for coping with this type of data. They aim at inducing hypotheses consisting of *clauses* c (abstract rules) such as

$$\texttt{localization(A)}\ :-\ \texttt{neighbour(A,B)},\texttt{localization(B)}$$

which consist of a head(c) $\equiv$ $\texttt{localization(A)}$ and a body(c) $\equiv$ $\{\texttt{neighbour(A,B)},\texttt{localization(B)}\}$. Upper-case strings denote *variables*, i.e., placeholders for objects. *Atoms* are predicates together with their arguments, for example $\texttt{localization(A,B)}$. A clause or atom is called *ground* if it does not contain any variables. Relational abstraction has two advantages: (1) variables such as $\texttt{A}$, i.e., placeholders for objects allow one to make abstraction of specific objects such as $\texttt{g1}$; (2) unification $\{\texttt{A/g3},\texttt{B/g245}\}$, i.e., the matching of variables allows one to evaluate abstract knowledge. Thus, relational learning allows to induce general regularities in terms of clauses but it does not handle uncertainty in a principled way. It is therefore not surprising that there has been a significant interest in integrating statistical learning with relational representations. This newly emerging research field is known under the name of *statistical relational learning* (SRL) and aims in principle at estimating a probability distribution $\mathbf{P}(\mathbf{X}, Y)$ over relational $\mathcal{X} \times \mathcal{Y}$. The key idea of SRL is to employ relational abstraction within statistical learning and therefore learning general (abstract) statistical regularities among groups of entities.

For classification, most SRL approaches (in particular the ILP motivated ones) are *generative*, i.e., they aim at estimating the joint distribution $\mathbf{P}(\mathbf{X}, Y)$ by learning the class prior distribution $\mathbf{P}(Y)$ and the class-conditional feature distribution $\mathbf{P}(\mathbf{X}|Y)$. The required posterior distribution $P(Y = y|\mathbf{X} = \mathbf{x})$ is then obtained using Bayes' rule yielding $f(\mathbf{x}) = \arg\max_{y_i \in \mathcal{Y}} P(\mathbf{X} = \mathbf{x}|Y = y_i, \boldsymbol{\lambda}^*) \cdot P(Y = y_i|\boldsymbol{\lambda}^*)$ as solution to the classification problem 1. Here, $\boldsymbol{\lambda}^*$ are the maximum likelihood parameters of the given generative model, which are typically estimated using the EM algorithm.

The classification performance of a generative approach, however, is often lower than that of a discriminative classifier, which estimates $f : \mathcal{X} \to \mathcal{Y}$ directly without representing the full joint probability distribution $\mathbf{P}(\mathbf{X}, Y)$. To improve the classification accuracy of generative models, different kernel functions have been proposed to make use of the good predictive performance of kernel methods such as support vector machine (SVM) [20]. A prominent representative of these kernel functions is the *Fisher kernel* [9]. The key idea there is to use the gradient of the log likelihood of the generative model with respect to its parameters as features. The motivation to use this feature space is that the gradient captures the generative process rather than just the posterior probabilities.

Fisher kernels have successfully been applied in many learning problems where the instances are described in terms of attribute-value vectors and for sequences of logical atoms [12]. So far, however, they have not been applied to relational data. Our main contribution is the definition of *relational Fisher kernels*, i.e., Fisher kernels derived from SRL models.

Definition 2 (Relational Fisher Kernel). *Relational Fisher kernels are the family of kernel functions k obtained by using the gradient $U_{\mathbf{x}} = \nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)$ of the log likelihood of a statistical relational model with respect to the model's parameters as features.*

We will experimentally show that the predictive accuracy of a SRL model can considerably be improved using Fisher kernels and SVMs. As showcase, we will focus on Bayesian logic programs [10] as SRL model but the idea applies naturally to any other SRL.

The outline of the paper is as follows. After discussing related work, we review Fisher Kernels in Section 3. In Section 4, we devise relational Fisher kernels based on Bayesian logic programs. Before concluding, we experimentally evaluate relational Fisher kernels in Section 5.

2 Related Work

Discriminative learning and kernels have only recently started to receive attention within SRL. To the best of our knowledge, [22, 24, 23, 21] are the only ones who aim at discriminative (probabilistic) models for structured data. In contrast to relational Fisher kernels, however, [22] and [21] do not explore kernel functions but gradient-based optimization of the conditional likelihood $\mathbf{P}(y|\mathbf{x})$. Taskar *et al.* [24] present a max-margin algorithm, where the structure in the input/output is modeled by a (relational) Markov network and not by a (relational) Bayesian network. In contrast to all these SRL approaches, relational Fisher kernels are easier to implement because gradient-based optimization techniques are typically already implemented for parameter estimation of SRL models. Recently, Landwehr *et al.* [13] (and related approaches) tightly integrated Naïve Bayes with ILP techniques focusing on discriminative objective functions such as conditional likelihood. The idea has been recently even generalized to learning simple relational kernels [14]. They do not consider fully generative models and no recursive dependencies.

Indeed, there has been a lot of interest in kernels for structured input/output spaces data in the kernel community, see e.g. [6, 25] and references in their. For structure input, there are in principle two ways to apply support vector machines to structured data: Using *syntax-driven* and *model-driven* kernel functions.

Syntax-driven kernels *decompose* the input 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 cross-product kernels is reflected in most work on kernels for structured data, from the early and influential technical reports [8, 28], through work on string kernels, kernels for higher order terms, and tree kernels, to more recent work on graph and relational kernels such as [3, 18]. They are not generative models.

An alternative to syntax-driven kernels are *model-driven kernels* like Fisher kernels. For instance [26] introduced the TOP kernel function, which is the scalar product between the posterior log-odds of the model and the gradient thereof. The posterior log-odds is the difference in the logarithm of the probability of each class given the instance. Marginalized kernels [27] have later been introduces as a generalization of Fisher kernels. Here, a kernel over both the hidden and the observed data is assumed. The marginalized kernel for the observed data is obtained by taking the expectation over the hidden variables. One advantage of *model-driven kernels* is their ability to explain the data using the underlying generative models. This is generally not the case for the recently proposed generalizations of the classical maximum-margin formulations to structured input/ouput spaces have been proposed, see e.g. [25] and references in their.

3 Kernel Methods and Probabilistic Models

Support vector machines [20] are one 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 'centered' at individual training examples. It is known that such functions can be interpreted 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 labeled 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.

Fisher kernels are 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. More precisely, given a parametric probability model M with parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, maximum likelihood parameters $\boldsymbol{\lambda}^*$, and output probability $P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}, M)$, the Fisher score mapping $U_\mathbf{x}$ is defined as $U_\mathbf{x} = \nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid$

$\boldsymbol{\lambda}^*, M) = (\{\partial \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\}/\partial\lambda_1, \ldots, \{\partial \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\}/\partial\lambda_n)^\top$ The Fisher information matrix is the expectation of the outer product of the Fisher scores over $P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}, M)$, more precisely, $J_{\boldsymbol{\lambda}} = E_{\mathbf{x}} \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{x} \mid \boldsymbol{\lambda}, M)\right] \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{x} \mid \boldsymbol{\lambda}, M)\right]^\top$. Given these definitions, the Fisher kernel is defined as $k(\mathbf{x}, \mathbf{x}') = U_{\mathbf{x}}^\top J_{\boldsymbol{\lambda}^*}^{-1} U_{\mathbf{x}'} =$

$$= \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\right]^\top J_{\boldsymbol{\lambda}^*}^{-1} \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x}' \mid \boldsymbol{\lambda}^*, M)\right]. \qquad (1)$$

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

4 Relational Fisher Kernels

To devise Fisher kernels for relational data, Equation (1) tells us that it is sufficient to compute the gradient of the log likelihood of a data case with respect to the parameters $\boldsymbol{\lambda}$ of any SRL model for relational data. This also explains the schematic nature of Definition 2 of relational Fisher kernels: *Any SRL model appropriate for the type of data at hand can be used to implement relational Fisher kernels.* Here, we will focus on Bayesian logic programs as SRL model, which we will briefly review now. For more information we refer to [11].

Bayesian Logic Programs [10, 11] (BLPs) integrate definite logic programs with Bayesian networks [19] and specify probability distributions over sets of ground atoms. The key idea is to view ground atoms as random variables, thus atoms describe groups of random variables. As an example, consider the KDD Cup BLP shown in Figure 1. The rule graph gives an overview of all probabilistic dependencies (black boxes) among abstract random variables (ovals). For instance, `interaction/1` is specified in terms of `neighbours/1` and `localization/1`. Each dependency gives rise to a local probabilistic model which is composed of a qualitative and a quantitative part. For instance, clause $C2$ `neighbours(GeneX) | neighbour(GeneX, GeneY), localization(GeneY)` in Figure 1 encodes that *"the neighbouring information depends on the localization of a neighbouring gene."* Gradient gray ovals represent abstract random variables such as `localization(GeneY)`, which take values from some domain D(`localization/2`). Smaller white circles on boundaries denote arguments, e.g., some genes `GeneY`. Larger white ovals together with undirected edges indicate that arguments refer to the same gene as for `localization(GeneX)` and `neighbour(GeneX, GeneY)`. To quantify the structural knowledge, *conditional probability distributions* $\mathrm{cpd}(c_i)$ are associated with clauses c_i. They encode the distribution of each possible value of the random variable in the head, given the values of the atoms in the body, i.e., $\mathrm{cpd}(c_i)_{jk} = P(u_j \mid \mathbf{u}_k)$, where $u_j \in \mathrm{D}(\mathrm{head}(c))$ and $\mathbf{u}_j \in \mathrm{D}(\mathrm{body}(c))$. Some information might be of qualitative nature only, such as `neighour(GeneX, GeneY)`. It does not affect the distribution but ensures the variable bindings among `neighbours(GeneX)` and `localization(GeneY)`. Such 'logical' atoms are solid gray ovals. Furthermore,

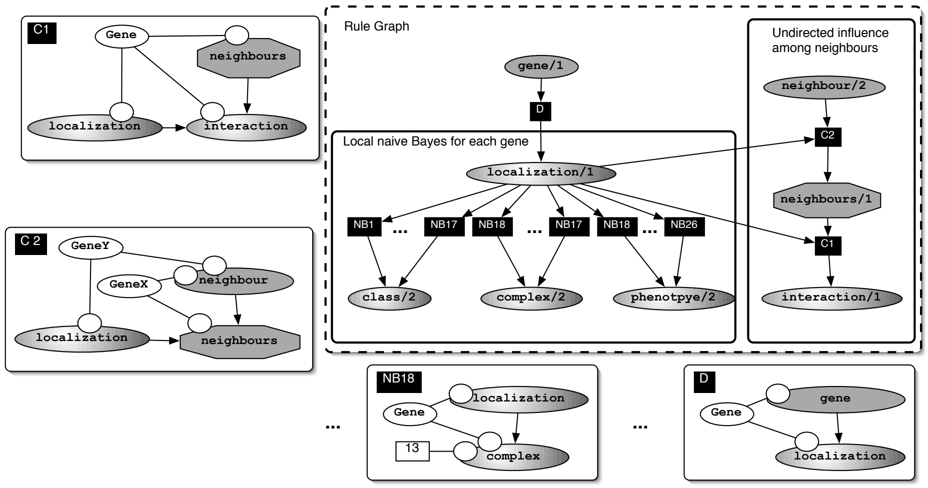

Fig. 1. *Localization* Bayesian logic program. The Bayesian clauses *NB1*, . . . , *NB17* encode a Naïve Bayes over local features of `Gene`. Clause *D* encodes the prior distribution over `localization` for each `Gene`. Clause *C2* aggregates the localizations of all neighbouring genes of `Gene` in `neighbours`(`Gene`). The *mode* of the localizations is used as aggregate function. Clause *C1* implements a mutual influence among `localization`(`Gene`) and the aggregated localization of interacting genes, `neighbours`(`Gene`).

octagonal such as `neighbours`(`GeneX`) denote *aggregation*, i.e., deterministic random variables that summarize the joint state of their parents into a singleton.

The semantics of a BLP M is defined in terms of a Bayesian network. Each ground instance $c\theta$ of a clause c in M, which is entailed by M ($M \models c$), constitute a node head($c\theta$) and its parent body($c\theta$) in the network. The distribution cpd(c) is associated with head($c\theta$) as conditional probability distribution. In case of multiple ground clauses with the same atom in the head, a *combining rule* such as *noisy or* or *mode* is used to combine the cpds associated with the node.

Kersting and De Raedt [10] have shown how to compute the gradient of a BLP w.r.t. a data cases. A data case D is set of (ground atom, state) pairs. The parameter vector $\boldsymbol{\lambda}$ of M consists of all $\text{cpd}(c_i)_{jk}$. Assuming decomposable combining rules, i.e., combining rules, which can be expressed in the structure of the induced Bayesian network (see [10]), the partial derivative of the log-likelihood with respect to a parameter λ of $\boldsymbol{\lambda}$ is

$$\frac{\partial \log P(D|\boldsymbol{\lambda}, M)}{\partial \lambda} = \sum_{\substack{\text{subst. } \theta \text{ with} \\ \text{support}(c_i\theta)}} \frac{P_N(\text{head}(c_i\theta) = u_j, \text{body}(c_i\theta) = \mathbf{u}_k \mid D)}{\text{cpd}(c_i\theta)_{jk}} \tag{2}$$

where P_N denotes the probability distribution of the Bayesian network induced by M for data case D. Note that, in contrast to parameter estimation, we do not reparameterize the Bayesian logic program.

In many cases, it is difficult — if not impossible — to devise a generative Bayesian logic program specifying a probability distribution, which sums up to one over all possible instances, say proteins. For example in our experiments, examples are partly specified within the logical background knowledge. Consequently, their probabilities do not sum up to one and Equation (2) is sensitive to the number of contributing ground clauses. Normalizing (2) with respect to the number of contributing ground clauses, i.e., to compute $\{|\{\theta|\,\mathrm{support}(c_i\theta)\}|\}^{-1} \cdot (\partial \log P(D|\boldsymbol{\lambda}, M)/\partial\lambda)$ worked well in our experiments.

5 Experimental Evaluation

The normalized version of Equation (2) is all we need to devise Fisher kernels for relational data such as the KDD cup 2001 data. In this section, we will experimentally evaluate them. Our intention here is to investigate to which extent relational Fisher kernels are competitive with the generative approach:

Q Do relational Fisher kernels considerably improve the predictive accuracies of their probabilistic baselines with plug-in estimates?

To investigate **Q**, we compare results achieved by Bayesian logic programs alone with results achieved by relational Fisher kernels based on Bayesian logic programs combined with SVMs. The experiments took place in two different domains: protein localization and web page classification. Both data sets are *collective* respectively *networked* data sets (see [16] and references in their), i.e., relational data where individual examples are interconnected, such as web pages (connected through hyperlinks) or gene (connected through interactions). This contrasts with traditional relational domains such as molecules where each individual example is a graph of connected parts. Traditionally, machine learning methods treat examples as independent, i.e., the classification task is treated as a local optimization problem. In contrast, within collective classification tasks, the class membership of one individual may have an influence on the class membership of a related individual. Thus, collective classification induces a global optimization problem.

There is a wide range of possible models that one can apply to the two tasks. We selected a set of models that we felt represented the main idea underlying a variety of collective learners [7, 15, 16] who globally combine local, propositional Naïve Bayes classifiers. Relational Fisher kernels based on Bayesian logic programs, however, are not designed for collective classification [1]. They assume each individual example as a graph of connected parts. Therefore, we apply the following trick. While learning in a collective way, we consider only individuals together with their direct neighbours at classification time, cf. Figure 2. For any individual without any neighbours, we used a copy of the individual as neighbour. Note that the direct neighbors can come from either the training or test set.

[1] Taking the whole graph at classification time would essentially yield the same feature vector for each individual because the data does not change.

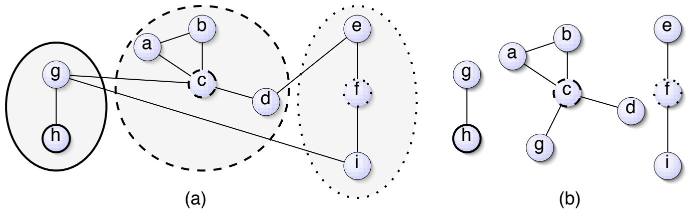

Fig. 2. (a) A collective data set, i.e., a graph of connected individuals each described by a set of local features. **(b)** Data set broken into subgraphs centered around individuals. Each subgraph consists of an individual and all its direct neighbours. Individuals can appear in multiple fragments such as g.

Therefore, their labels are either known or unknown, respectively. This is akin to *iterative* classifiers (see e.g. [16]), which also treat each individual together with all its direct neighbours as a single data case.

We investigated collective Naïve Bayes models and relational Fisher kernels derived from them as described above together with SVMs. We used Weka's [29] using polynomial kernels. To reduce the number of features of the local Naïve Bayes models, we performed WEKA's greedy subset evaluation with default parameters on the training set. That is, we start with an empty feature set for the Naïve Bayes and add one feature on each iteration. If we have added all features or there is no improvement in score of the Naïve Bayes from adding any further features, the search stops and returns the current set of features. To score feature subsets, we used 10-fold cross-validated classification accuracy of the Naïve Bayes on the training set. Finally, both classification tasks are multiclass problems. In order to tackle multiclass problems with SVMs, we followed a round robin approach [5]. That is, each pair of classes is treated as a separate classification problem. The overall classification of an example instance is the majority vote among all pairwise classification problems.

Protein Localization Reconsider the KDD Cup 2001 localization task of example 1. Figure 1 shows the Bayesian logic program used in the experiments. We listed the genes as ground atoms over `gene`/1 in the logical background knowledge. They were used to encode the prior localization, cf. Bayesian clause D. The feature selection yielded 26 features for the local Naïve Bayes describing the genes, which we encoded as Bayesian clauses $NB1, \ldots, NB26$. So far, the Bayesian logic program encodes the simple, non-collective Naïve Bayes model we used in the experiments. To model the collective nature of the data set, we enriched the Naïve Bayes model as follows. We encoded each interaction as a logical ground atom over `d_neighbour`/2, i.e., we omitted the originally given quantification of the interactions. Because interactions are bidirectional, i.e., undirected, we additionally defined the symmetric closure `neighbour(A,B) :- d_neighbour(A,B);d_neighbour(B,A)` (where ';' denotes a logical or) as logical background. The localizations of the direct neighbours of

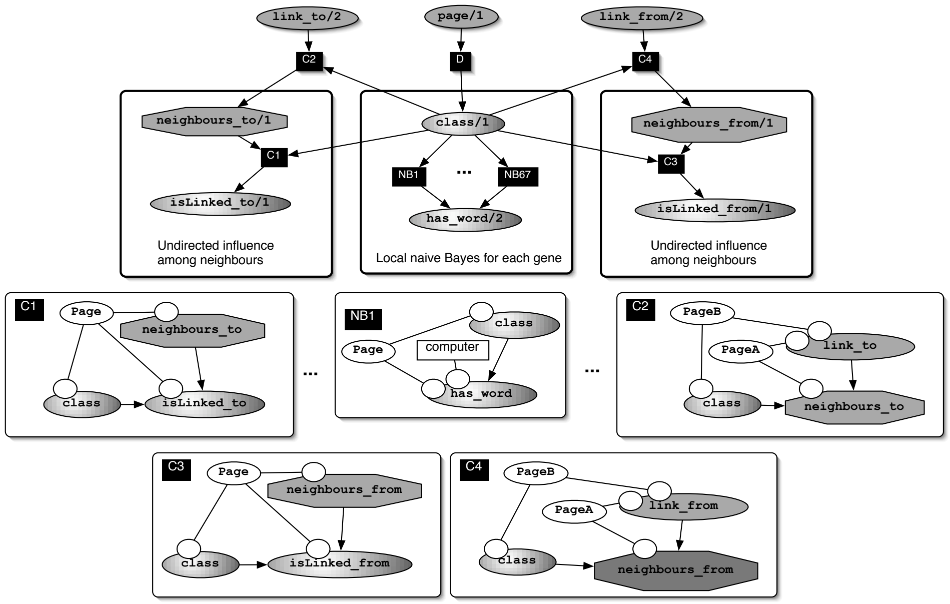

Fig. 3. *WebKB* Bayesian logic program. The Bayesian clauses *NB1*, . . . , *NB67* encode a Naïve Bayes over local features of genes web pages, `page`(`Page`). Clause *D* encodes the prior distribution over `class` for each `Page`. Clause *C2* aggregates the class memberships of all web pages to which `Page` provides a link. Clause *C4* aggregates the class memberships of all web pages, which link to `Page`. In both cases, the *mode* of the class memberships is used as aggregate function. Clauses *C1* and *C3* implement a mutual influence among `class`(`Page`) and the aggregated class memberships of linked pages.

a `Gene` are aggregated in clause *C2* into a single value `neighbours`(`Gene`) using the *mode* of the interactions. To establish a mutual influence among the localizations of a gene and its neighbours, we introduced a boolean random variable `interaction`(`Gene`), which has `neighbours`(`Gene`) and `localization`(`Gene`) as parents, cf. clause *C1*. Setting the evidence for `interaction`(`Gene`) always to be `true` guarantees that both parents are never d-separated, hence, they are probabilistic dependent.

On the test set, the relational Fisher kernel achieved an accuracy of 72.89%, whereas the collective Naïve Bayes only achieved 61.66%, and outperformed Hayashi et al.'s KDD Cup 2001 winning nearest-neighbour approach [1] that achieved a test set accuracy of 72.18%. This affirmatively answers **Q**.

Web Page Classification This dataset is based on the WebKB Project [2]. It consists of sets of web pages from four CS departments, with each page manually labeled into 7 categories: course, department, faculty, project, staff, student or others. We excluded pages in the 'other' category from consideration and put

	Cornell	Texas	Washington	Wisconsin	**Mean**
Collective Naïve Bayes	$63,44$	$59,20$	$58,65$	$68,07$	$62,34$
Relational Fisher Kernel	$71,08$	$73,53$	$71,93$	$84,59$	$75,28$

Table 1. Leave-one-university-out accuracies on the WebKB data. Both collective classifiers used the same Bayesian logic program. The mean difference between collective Naïve Bayes and relational Fisher kernel in test accuracy was 12.94%.

them into the background knowledge. This yielded a multiclass problem with 6 different classes, 877 web pages, and 1516 links among the web pages.

Figure 3 shows the Bayesian logic program used in the experiments. It essentially follows the idea underlying the Bayesian logic program for the localization task, cf. Figure 1. The feature selection yielded 67 local for the local Naïve Bayes model (clauses *NB1*, …, *NB67*. Whereas gene interaction is undirected, links among web pages are directed. There are *incoming* and *outcoming* links on a web page. We modeled their influences on the class membership of a web pages separately. The atom `neighbours_from(Page)` (respectively `neighbours_to(Page)`) aggregates the class memberships of all pages that have a link to `Page` (respectively that `Page` links to) using *mode* as aggregate function. Again, we took care that `class(Page)` and the aggregated class memberships of linked pages mutually influence each other, i.e., we introduced `isLinked_from(Page)` and `isLinked_to(page)`, whose evidence is always `yes`.

We performed a leave-one-university-out cross-validation. The experimental results are summarized in Table 1. The Fisher kernels achieved an accuracy of 75.28%, which is significantly higher (two-tailed t-test, $p = 0.05$) than the collective Naïve Bayes' accuracy of 62.34%. For comparison, the performance of the collective Naïve Bayes is in the range of Getoor *et al.*'s [7] probabilistic relational model with link anchor words. The Fisher kernel outperforms the probabilistic relational model with the best predictive accuracy Getoor *et al.* report on. It takes structural uncertainty over the link relationship of web pages into account and achieved with 68% its highest accuracy on the Washington hold-out set. Thus, **Q** is again affirmatively answered.

6 Conclusions and Future Work

In this paper, Fisher kernels for relational data have been introduced and experimentally investigated. They are 'off-the-shelf' kernels and are easy to implement for any SRL model. The experimental results show that Fisher kernels can handle relational data and can indeed significantly improve the predictive performance of their underlying probabilistic model: the WebKB model is outperformed by an advanced probabilistic relational model, which in turn was outperformed by our Fisher kernel; the probabilistic KDD Cup model ranks only around the top 50% level of submitted models (61% accuracy) whereas the corresponding Fisher kernel performs better than the KDD Cup 2001 winning approach.

The research on the intersection of kernel, discriminative, and relational learning has just started, and relational Fisher kernels are only a further step into this direction. There is a lot of space for future research: other learning tasks such as regression, clustering, and ranking should investigated. In general, choosing the appropriate kernel is the major step for the application of kernel method and should take as much domain knowledge into account as possible. To this aim, knowledge-based SVMs [4] have been for instance proposed, which find in addition to a large margin solution an estimate statisfying constraints encoding prior knoweldge in terms of polyhedral sets. As [3] point out, real-valued functions are inappropriate as a general knowledge representation language; they suffer from a non-declarative nature. Statistical relational languages are a natural alternative and an attractive way to embed knowledge into statistical learning algorithms in a principled and flexible way.

Acknowledgments The authors thank Luc De Raedt for his support. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic ILP II.*

References

1. J. Cheng, C. Hatzis, M.–A. Krogel, S. Morishita, D. Page, and J. Sese. KDD Cup 2001 Report. *SIGKDD Explorations*, 3(2):47 – 64, 2002.
2. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence Journal*, 118(1–2):69–113, 2000.
3. P. Frasconi, A. Passerini, S. H. Muggleton, and H. Lodhi. Declarative kernels. Submitted, 2005.
4. G. Fung, O. Mangasaruan, and J. Shavlik. Knowledge-based Support Vector Machine Classifier. In *Advances in Neural Information Processing Systems 15*, 2002.
5. J. Fürnkranz. Round Robin Classification. *Journal of Machine Learning Research (JMLR)*, 2:721–747, 2002.
6. T. Gärtner. Kernel-based Learning in Multi-Relational Data Mining. *ACM-SIGKDD Explorations*, 5(1):49–58, 2003.
7. L. Getoor, N.Friedman, D. Koller, and B Taskar. Learning Probabilistic Models of Link Structure. *Journal of Machine Leaning Research (JMLR)*, 3:679 – 707, 2002.
8. D. Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
9. T. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493, 1999.
10. K. Kersting and L. De Raedt. Adaptive Bayesian Logic Programs. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-01)*, volume 2157 of *LNAI*, pages 118–131, Strasbourg, France, September 2001. Springer.
11. K. Kersting and L. De Raedt. Bayesian Logic Programming: Theory and Tool. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning.* MIT Press, 2006. (to appear).

12. K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning (ECML-04)*, volume 3201 of *LNCS*, pages 205 – 216, Pisa, Italy, 2004.
13. N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naïve Bayes and Foil. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 795–800, Pittsburgh, Pennsylvania, USA, July 9–13 2005. AAAI.
14. N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. kFOIL: Learning Simple Relational Kernels. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*. AAAI, 2006. (To appear).
15. Q. Lu and L. Getoor. Link-based Classification. In T. Fawcett and N. Mishra, editors, *Proceedings of the International Conference on Machine Learning (ICML-03)*, pages 496–503, Washington, DC USA, August 21-24 2003.
16. S. A. Macskassy and F. Provost. Classification in Networked Data: A toolkit and a univariate case study. Technical Report CeDER-04-08, CeDER Working Paper, Stern School of Business, New York University, New York, USA, 2004.
17. S. H Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19(20):629–679, 1994.
18. A. Passerini, P. Frasconi, and L. De Raedt. Kernels on Prolog Proof Trees: Statistical Learning in the ILP Setting. *JMLR*, 7:307–342, 2006.
19. J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition, 1991.
20. B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002.
21. P. Singla and P. Domingos. Discriminative training of markov logic networks. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 868–873, Pittsburgh, Pennsylvania, USA, July 9–13 2005. AAAI Press.
22. B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In A. Darwiche and N. Friedman, editors, *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, Edmonton, Alberta, Canada, August 1-4 2002.
23. B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning Structured Prediction Models: A Large Margin Approach. In L. De Raedt and S. Wrobel, editors, *Proceedings of the Twenty Second International Conference on Machine Learning (ICML-05)*, pages 897–902, Bonn, Germany, August 7-10 2005.
24. B. Taskar, C. Guestrin, and D. Koller. Max-Margin Networks. In *Advances in Neural Information Processing Systems 16*, 2004.
25. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.
26. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 977–984. The MIT Press, 2002.
27. K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 2002.
28. C. Watkins. Kernels from matching operations. Technical report, Department of Computer Science, Royal Holloway, University of London, 1999.
29. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.