Support Vector Inductive Logic Programming

Stephen Muggleton[1], Huma Lodhi[1],
Ata Amini[2], and Michael J. E. Sternberg[2]

[1] Department of Computing, Imperial College,
180 Queen's Gate, London, SW7 2AZ, UK
{shm, hml}@doc.ic.ac.uk
[2] Department of Biological Sciences, Imperial College,
180 Queen's Gate, London, SW7 2AZ, UK
{ata.amini, m.sternberg}@imperial.ac.uk

Abstract. In this paper we explore a topic which is at the intersection of two areas of Machine Learning: namely Support Vector Machines (SVMs) and Inductive Logic Programming (ILP). We propose a general method for constructing kernels for Support Vector Inductive Logic Programming (SVILP). The kernel not only captures the semantic and syntactic relational information contained in the data but also provides the flexibility of using arbitrary forms of structured and non-structured data coded in a relational way. While specialised kernels have been developed for strings, trees and graphs our approach uses declarative background knowledge to provide the learning bias. The use of explicitly encoded background knowledge distinguishes SVILP from existing relational kernels which in ILP-terms work purely at the atomic generalisation level. The SVILP approach is a form of generalisation relative to background knowledge, though the final combining function for the ILP-learned clauses is an SVM rather than a logical conjunction. We evaluate SVILP empirically against related approaches, including an industry-standard toxin predictor called TOPKAT. Evaluation is conducted on a new broad-ranging toxicity dataset (DSSTox). The experimental results demonstrate that our approach significantly outperforms all other approaches in the study.

1 Introduction

In this paper we propose a novel machine learning approach which combines the dimensionality independence advantages of Support Vector Machines (SVMs) with the expressive power and flexibility of Inductive Logic Programming (ILP). In particular, we propose a kernel which is an inner product in the feature space spanned by a given set of first order hypothesised clauses. As with normal ILP, examples, background knowledge and hypothesised clauses are encoded as logic programs. The kernel not only captures the semantic and syntactic relational information contained in the data but also provides the flexibility of using arbitrary forms of structured and non-structured data.

The approach we suggest differs from the relational kernels suggested in [1,2] by our use of logical background knowledge. In order to understand the distinction being made here consider the following three settings for ILP.

Atomic Generalisation. This setting is characterised by having examples of which are typically ground atomic formulae and hypotheses consisting of atomic formulae which

A. Hoffmann, H. Motoda, and T. Scheffer (Eds.): DS 2005, LNAI 3735, pp. 163–175, 2005.

entail the examples. Plotkin [3] showed that this hypothesis space forms a lattice which is partially ordered by atomic subsumption.

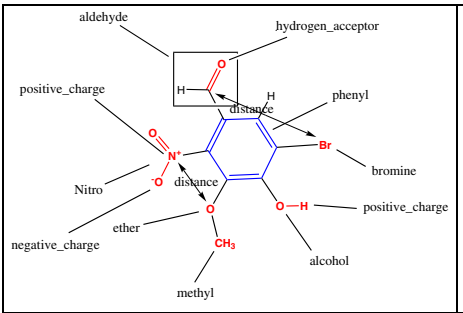
Clausal Generalisation. In this setting examples are ground clauses and hypotheses are clauses which entail the examples. Plotkin [4] showed that once more this hypothesis space forms a lattice which is partially ordered by clausal subsumption.

Clausal Generalisation Relative to Background Knowledge. This third setting [4] is distinguished by assuming the existence of background knowledge in the form of a conjunction of clauses. Examples are ground clauses. Hypotheses are clauses which when conjoined with the background knowledge entail the examples.

Most ILP research has assumed the third setting, clausal generalisation relative to background knowledge, since this is the more general approach. The use of background knowledge provides a flexible way of encoding the understanding of domain experts, and can increase both the predictive accuracy of the learning and the degree of insight provided relative to the background knowledge. However, this setting brings with it overheads related to the theorem proving involved in using background knowledge. For this reason Page and Frisch [5] investigated the use of atomic generalisation with respect to a monadic constraint theory. This is a generalisation of the first setting, and a special case of the third setting.

More recently Lloyd [6] and others [2] have investigated algorithms which use the setting of atomic generalisation, but with more general forms of strongly-typed terms. In particular, terms can consist of arbitrary sets. This allows more flexibility for defining data types without the overheads associated with background knowledge. In [2] it is shown that this form of representation and learning can be used to formulate a relational kernel. In [1] it is shown that by using the "bag of atoms" representation introduced in [7] a multi-instance kernel approach can even be applied to structurally complex ILP learning problems involving small molecules.

The SVILP approach is a form of generalisation relative to background knowledge, though the final combining function for the ILP-learned clauses is an SVM rather than a logical conjunction. We will now provide a simplified worked example to show the difference in representing molecules using the Gärtner/Chevaleyre approach from the representation used by our SVILP kernel. Figure 1 shows a typical molecule from the DSSTox dataset of toxins (see Section 5). In the SVILP approach we start by formulating chemical background knowledge in the form of Prolog definitions. These have been designed by one of the authors (Ata Amini), a biochemistry domain expert, to be relevant to properties associated with toxins. Such properties include the existence of substructures such as aromatic rings, methyl and alcohol substructures, types of atom, charge, the existence of hydrogen acceptors and distances between various critical structures on the molecule. The ILP system CProgol5.0 is used to generate a set of hypothesised clauses based on the given background knowledge and examples. An SVM kernel is then used as the combining function for predictions of these clauses. By contrast, the Gärtner/Chevaleyre features consist simply of the frequency of occurrence of atoms, bonds and atom pairs within the given molecule. These are used to form a vector representation of the molecule. An obvious advantage of this "bag-of-atoms" representation is that it requires no domain expertise and thus is less effort to develop. By analogy with



Br: 1
C: 8
H: 6
N: 1
O: 5
Single Bonds: 16
Double and Aromatic Bonds: 5

a) SVILP representation | b) Gärtner/Chevaleyre representation

Fig. 1. Molecule represented using a) SVILP representation which employs a kernel based on domain-expert informed chemical background knowledge indicated by the annotations on the figure and b) Gärtner/Chevaleyre bag-of-atoms uses Multi-Instance (MI) kernel based on frequency of occurrences of atoms and atom pairs

the use of the "bag-of-words" [8] representation in text classification one might expect a simple representation of this form to lead to superior predictive accuracy. However, this is not the case in the experiments reported in Section 5 in which the SVILP kernel significantly outperforms the Gärtner/Chevaleyre kernel. In this case, the use of more highly informed background knowledge in the SVILP appears to provide a significant advantage.

The paper is arranged as follows. The Background Section 2 introduces the basic ideas behind kernels, SVMs and Inductive Logic Programming (ILP). In Section 3 SVILPs are defined and their properties proved. This is followed by a section which describes Related Work (Section 4). Next we describe the Experiments (Section 5) on toxicity data. The paper then concludes.

2 Background

Kernels and Support Vector Machines: During recent years, there has been increasing interest in kernel-based methods such as Support Vector Machines (SVMs) [9]. The non-dependence of these methods on the dimensionality of the feature space and the flexibility of using any kernel make them a good choice for different tasks such as classification and regression. We can view the learning process of SVMs as comprising two stages. 1) Map the input data, $d_1, \ldots, d_n \in D$, into some higher dimensional space H through a non-linear mapping ϕ that is given by $\phi : D \rightarrow H$. The mapping ϕ may not be known explicitly but be accessed via the kernel function described below. 2) Construct a linear function f in the space.

The kernel function $K(d_i, d_j) = \langle \phi(d_i), \phi(d_j) \rangle$ computes the inner product between the mapped instances. The mathematical foundation of such a function was established during the first decade of the twentieth century [10]. A kernel function is a symmetric function, $K(d_i, d_j) = K(d_j, d_i) \quad \text{for } i, j = 1, \ldots, n$, and satisfies the property of positive semi-definiteness, $\sum_{1,j=1}^{n} a_i a_j K(d_i, d_j) \geq 0 \quad \text{for } a_i, a_j \in R$.

The $n \times n$ matrix with entries of the form $K_{ij} = K(d_i, d_j)$ is known as the kernel matrix or the Gram matrix. A kernel matrix is a symmetric, positive definite matrix. In other words the n Eigen values of this $n \times n$ kernel matrix are non-negative. Kernel functions can be defined over general sets [11]. This important fact has allowed successful exploration of novel kernels for discrete spaces such as strings and graphs [12,13].

Inductive Logic Programming: Inductive Logic Programming (ILP) [14] is the area of AI which deals with the induction of hypothesised predicate definitions. In ILP logic programs are used as a single representation for examples, background knowledge and hypotheses. ILP is differentiated from most other forms of Machine Learning (ML) both by its use of an expressive representation language and its ability to make use of logically encoded background knowledge. This has allowed successful applications of ILP in areas such as molecular biology [15] and chemoinformatics [16].

In the following it is assumed that the examples, background knowledge and hypotheses each consist of logic programs, ie sets of first-order Horn clauses. The normal semantics of ILP is as follows. We are given background (prior) knowledge B and evidence E. The evidence $E = E^+ \wedge E^-$ consists of positive evidence E^+ and negative evidence E^-. The aim is then to find a hypothesis H such that the following conditions hold.

Prior Satisfiability. $B \wedge E^- \not\models$
Posterior Satisfiability. $B \wedge H \wedge E^- \not\models$
Prior Necessity. $B \not\models E^+$
Posterior Sufficiency. $B \wedge H \models E^+$

Since a large number of hypotheses will typically fit such a definition, the Bayesian ILP setting [17] assumes a prior probability distribution defined over the hypothesis space. Algorithms such as CProgol [18] use such a prior to search for hypotheses which maximise the posterior probability $p(H|E)$.

3 Support Vector Inductive Logic Programming (SVILP)

The SVILP framework builds on the ILP framework. Thus we also assume background knowledge B, examples E and a hypothesis H for which the conditions of the normal semantics hold. The key difference between ILP and SVILP is the way in which the set of clauses H is used for predictive purposes. In ILP H is simply treated as a conjunction, for which any instance d from the domain of instances D is predicted to be true if and only if $B, H \models d$.

By contrast, SVILP bases a kernel on the predictions of the clauses h in H. This involves forming a binary hypothesis-instance association matrix M in which element $M_{ij} = 1$ (0 otherwise) if and only if clause $h_i \in H$ entails instance $d_j \in D$ as follows, $B, h_i \models d_j$.

The kernel described in Section 3.2 can be viewed as a function for which similarity of two instances d_1 and d_2 is based on the similarity of the rows of clauses in M associated with d_1 and d_2.

father(henry,john). father(david,henry). mother(jane,john). mother(elizabeth,henry).
father(charles,mary). father(egbert,jill). mother(jill,mary). mother(ann,jill).

grandfather(F,P) ← father(F,P1), parent(P1,P).
grandmother(M,P) ← mother(M,P1), parent(P1,P).
parent(F,P) ← father(F,P).
parent(M,P) ← mother(M,P).

hair(john,blond). hair(mary,black).
hair(henry,blond). hair(charles,black).
hair(jill,blond). hair(elizabeth,blond).
hair(egbert,black). hair(ann,blond).
hair(david,black). hair(jane,black).

Fig. 2. Background knowledge for disease inheritance

3.1 Family Example

In this artificial example we assume that the occurrence of a disease is related to the inheritance patterns of an observable property (e.g., hair colour) in various families. The background knowledge is shown in Figure 2. This describes the relationships in the family tree shown in Figure 6. Examples of individuals having the disease are shown in Figure 3 and various hypothesised clauses are shown in Figure 4. Assuming the domain is limited to the examples, we show the resulting binary hypothesis-instance association matrix in Figure 5.

1. disease(john) 2. disease(mary) 3. disease(jane) 4. disease(henry) 5. disease(charles)

Fig. 3. Examples for disease inheritance

A. disease(P) ← hair(P,Colour), father(F,P), hair(F,Colour)
B. disease(P) ← hair(P,Colour), mother(M,P), hair(M,Colour)
C. disease(P) ← hair(P,Colour), grandmother(M,P), hair(M,Colour)
D. disease(P) ← hair(P,Colour), grandfather(F,P), hair(F,Colour)
E. disease(P) ← hair(P,black), father(F,P), mother(M,P),
hair(F,blond), hair(M,black)
F. disease(P) ← hair(P,black), father(F,P), grandfather(G,P),
hair(F,blond), hair(G,black)

Fig. 4. Hypothesised clauses for disease inheritance

	A	B	C	D	E	F
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	1	0	0	0	0	0
4	0	1	0	0	0	0
5	1	0	0	0	0	0

Fig. 5. Resulting binary hypothesis-instance association matrix

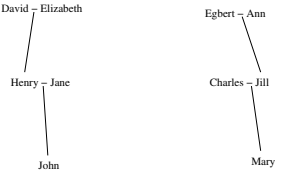


Fig. 6. Family trees for disease inheritance

Note that according to the matrix examples 1 and 2 have maximum similarity. This is despite the fact that the hair colour (the main observable feature) of John and Mary (the individuals involved in the examples) are opposite (blond and black respectively). The example demonstrates the strong learning bias which can be introduced by the use of background knowledge and hypotheses within the SVILP setting. In the next section we define the kernel formally.

3.2 Definition of Kernel

We assume background knowledge B and a set of hypothesised clauses H drawn from a class of hypotheses $\mathcal{H}$ and a set of instances D drawn from a class of instances $\mathcal{D}$. Each hypothesis clause h in H can be thought of as a function of the following form, $h : \mathcal{D} \rightarrow \{\text{True}, \text{False}\}$. Conversely the τ function gives the hypothesised clauses covering any particular instance, $\tau : \mathcal{D} \rightarrow 2^H$. Where for any d_i in D

$$\tau(d_i) = \{h : \exists h \in H, (B, h \models d_i)\}$$

As in the Bayesian ILP framework [17], we assume a prior probability distribution over the hypotheses. This can be represented as a function π such that

$$\pi : H \rightarrow [0, 1] \quad \text{and} \quad \sum_{h \in H} \pi(h) = 1$$

Next we define a function, which maps sets of hypothesised clauses to probabilities.

$$f : 2^H \rightarrow [0, 1]$$

For all $H' \subseteq H$

$$f(H') = \sum_{h \in H'} \pi(h)$$

Now the kernel function is as follows. For all d_i, d_j in D

$$K(d_i, d_j) = f(\tau(d_i) \cap \tau(d_j))$$

It can be easily shown that the kernel is an inner product in ILP space.The kernel requires a hypothesised clause set H. In order to improve the informative power of the kernel we define a prior probability distribution and fits the prior to the coordinates in space spanned by the hypothesised clauses. In this way a countable set of hypothesised clauses implies a mapping ϕ that maps the data into an ILP space, where dimensionality of the space is the same as the cardinality of the set of hypothesised clauses and each mapped instance can have r number of non-zero entries (in a column vector) where r is in the range $1 \leq r \leq k$. Formally

$$f_i(d) = \sqrt{\pi(h_i(d))} \quad \text{for } i = 1, \ldots, k$$

Hence the mapping ϕ for an instance is given by

$$\phi : d \rightarrow ((f_1(d), f_2(d), \ldots, f_k(d)) = (f_i(d)_{i=1}^k)$$

and kernel for instance d_i and d_j is given by

$$K(d_i, d_j) = \langle \phi(d_i), \phi(d_j) \rangle = \sum_{i=1}^{k} f_i(d_i) f_i(d_j)$$

Hence, $K(d_i, d_j) = f(\tau(d_i) \cap \tau(d_j))$

The validity of kernel function follows from the definition as an inner product however we can show that it satisfies Mercer's condition (symmetry and positive semi-definiteness). Clearly the kernel function is symmetric and positive semi-definiteness occurs since there is mapping ϕ from D into an ILP space. For all $a_i \in R$ and $d_i \in D$, for $i = 1, \ldots, n$ we have the following expression, $\sum_{1,j=1}^{n} a_i a_j K(d_i, d_j)$. We now use a compact representation $A = (a_i)_{i=1}^{n}$ and $\phi = (\phi(d_i))_{i=1}^{n}$, hence kernel matrix $\sum_{i,j=1}^{n} K(d_i, d_j) = \phi\phi'$ and the expression is, $A\phi'\phi A' = t't \geq 0$.

Given that ϕ maps the data into ILP space, we can construct Gaussian RBF kernels in ILP space $K_{RBF}(d_i, d_j) = \exp\left(\frac{-\|(\phi(d_i) - \phi(d_j)\|^2}{2\sigma^2}\right)$, where $\|(\phi(d_i) - \phi(d_j)\| = \sqrt{K(d_i, d_i) - 2K(d_i, d_j) + K(d_j, d_j)}$. Our method is flexible to construct any kernel in the space spanned by the clauses. However we select RBF kernels (K_{RBF}) constructed in ILP space for our experiments in section 5.

We now consider the analysis of the complexity of the kernel. Assuming the theorem prover can test each hypothesised clause against each instance in time bounded by a constant k, the overall time taken to compute the kernel is proportional to the number of hypothesised clauses $|H|$ and the number of instances $|D|$.

4 Related Work

Propositionalisation: Within ILP "propositionalisation" [19] techniques transform machine learning problems from a first-order logic setting into one which can be handled by a "propositional" or feature-based learner. Kramer et al. [19] distinguish between domain-independent ([20,21,22,23]) and domain-dependent approaches (eg [24]). In most domain-independent propositionalisation approaches [21,22,23] features are introduced as clauses with a monadic head predicate. For instance, when applied to problems involving molecular descriptions these techniques introduce new features such as the following.

```
f1(A):-has_rings(A, [R1, R2]), hydrophobic(A,H), H > 1.0.
f2(A):-atm(A,B,_,27,_), bond(A,B,C,_), atm(A,C,_,29,_).
```

Though superficially similar to domain-independent propositionalisation, the SVILP approach described in this paper is not a propositionalisation technique since it does not transform the representation by the introduction of such monadic features. Instead a general-purpose ILP learning algorithm is used to learn clauses with heads having arbitrary predicate arities. The heads of these clauses can contain terms with multi-arity function symbols and constants. In normal ILP the hypothesis used for predictive purposes would consist of these clauses conjoined together. In SVILP the truth-value predictions of these individual clauses are projected onto the instance space. The

kernel matrix is then formulated over the instance-space predictions of the individual clauses.

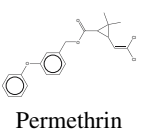
SVILP is similar in its use of support-vector technology to the domain-dependent propositionalisation approach of Kramer and Frank [24]. This uses bottom-up evaluation to fine. The key difference here is that SVILP is domain-independent, allowing the use of background knowledge to encode the appropriate machine learning bias.

Kernels within ILP: Within ILP there has recently been interest in the development of kernels which incorporate relational information, for use within support vector machines[2,25,26,27]. Several authors [2,25] take the approach of using syntactic measures of distance between first-order formulae as the basis for such kernels. Within the ILP literature it is normal to differentiate between *syntactic* [28,29] and *semantic* [30] distance measures. Syntactic measures are based on differences in the structure of first-order formulae, and tend to be confined to comparison of terms, rather than arbitrary first-order formulae. Semantic measures are based on comparison of models, making this approach intractable for all but simple formulae.

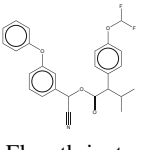
The kernel approaches described in [2,25] are unable to make use of background knowledge, since they are based on syntactic comparison of ground atoms. By contrast, a central feature of the SVILP described in this paper is its use of generalisation relative to background knowledge.

5 Experiments

A new dataset was used for evaluating SVILP. The DSSTox dataset was made available to us by Dr Ann Richards of National Health and Environmental Effects Research Laboratory, USA. The dataset represents the most diverse set of toxins presently available in the public domain. By choosing a new toxin dataset we avoided over-testing problems associated with molecular datasets such as the Mutagens [31]. The 188 molecule Mutagenic dataset has now been evaluated by so many researchers that it is becoming hard to argue that some of the higher reported accuracies are not simply due to chance.



Permethrin



Flucythrinate

Fig. 7. Examples of compounds in DSSTox

Materials: The DSSTox [32] database contains organic and organometalic molecules with their toxicity values. The dataset consists of 576 molecules. Figure 7 shows an example of two of the molecules found in DSSTox. As far as we know no previous attempt has been made to quantify the structure and activity relationship for the whole DSSTox dataset.

Methods: We now describe the pre-processing stage. Molecules in the form of SMILES strings, were transformed into 3D structures using the software CONCORD 4.0 [33] (implemented in TRIPOS). All of the molecules contain continuous chemical feature known as the lowest unoccupied molecule orbital (LUMO), water/octanol partition coefficient (LOGP) and dipole moment. LOGP reflects the hydrophobicity of compounds and the mechanism of toxicities of these chemicals are based on their accumulation in the non-polar lipid phase of the biomembranes. LUMO and dipole moment can describe electrophilicities of compounds. The key information is given in the form of atom and bond description.

We compared the performance of SVILP with a number of related techniques including partial least squares (PLS), multi instance kernels (MIK) [1,2], an RBF kernel using only 3 chemical features (LOGP, LUMO, dipole moment) that we term as CHEM. We also compared the performance of SVILP with well known QSAR software TOPKAT (Toxicity Prediction by Komputer Assisted Technology).

As our experimental methodology we used 5-fold cross validation. For evaluation we used mean squared error (MSE) and R-squared (standard measure of accuracy in QSAR). In this work we employed $\epsilon-$insensitive SVM regression (SVR)[9]. We used the SVM package SVMTorch [34] for our experiments. C (regularization parameter), ϵ (controls width of insensitive band), σ (width of Gaussian) are the tunable parameters for kernel-based methods (SVILP, CHEM and MIK). In PLS the tunable parameter is the "number of components". These parameters can be set by some model selection method. The traditional protocol to set the values for the parameters is the minimisation (maximisation) of some criterion relative to the values of the parameters using a validation set. We select the optimal values of the tunable parameters using a validation set as described. We set the parameters for each fold using only the training set of the fold. We randomly selected a subset comprising 75% of the data (training set of each fold) for the training set and used the remaining data as a test set. A range of values of the parameters were selected. The sets of the values are given by $C = \{10, 100, 1000, 10000\}$, $\epsilon = \{0.1, 0.3, 0.5, 1.0\}, \sigma = \{0.125, 0.25, 0.5, 4, 16\}$. For PLS we used the number of components from 1 to 15. The parameters which give the minimum MSE on the validation set were chosen. For the selected parameters we obtained the models (created by the methods) using full training set and performed evaluation on test compounds.

In order to perform the prediction task using SVILP, we first obtained a set of clauses. Examples and Background knowledge (atom-bond, high level chemical groups e.g. phenyl ring, aldehyde, carboxylic acids and chemical features) are given to CProgol5.0 [18] which generates a set of hypothesised clauses. For all the folds, the clauses with positive compression were selected where the number of obtained clauses for each fold can vary between 1500-2000. The compression value of a clause is given by $V = \frac{P*(p-(n+c+h))}{p}$, where p is the number of positive instances correctly deducible from the clause, n is the number of negative examples incorrectly deducible from the clause, c is the length of the clause and h is number of further atoms to complete the input/output connectivity of the clause and P is the total number of positive examples. The hypothesised clauses are then taken by a Prolog program which computes the hypothesis-instance association (see Section 3), indicating for each instance the set of all hypothesised clauses which imply it. In this work we used a uniform probability

	MSE	R-squared
CHEM	0.811	0.519
PLS	0.671	0.593
MIK	0.838	0.503
SVILP	**0.574**	**0.655**

Fig. 8. MSE and R-squared for CHEM, PLS, MIK and SVILP

	Accuracy
ILP (CProgol5.0)	55
CHEM	58
PLS	71
MIK	60
SVILP	**73**

Fig. 9. Accuracy for ILP, CHEM, PLS, MIK and SVILP

distribution over the clauses. We then computed the similarity between molecules using proposed kernel. In order to apply PLS for toxicity prediction, we used the same set of hypothesised clauses generated by CProgol5.0 as SVILP.

Results: We conducted a series of experiments to evaluate the performance of the proposed method. We conducted the first set of experiments to evaluate the efficacy of the new method for predicting the toxicity values. Figure 8 shows the results. The results are averaged over 5 runs of the methods. Based on the statistical sign test method, SVILP shows significant improvement in comparison with the other methods in the study. In the second set of experiments we assessed the performance of the methods for qualitative prediction. We evaluated our approach by employing it for categorising the molecules into two categories, toxic and non-toxic. We also compared the performance of SVILP with the standard ILP system CProgol5.0. All of the methods predict the non-toxic molecules with high accuracy. Figure 9 shows the results for the category "toxic". According to McNemar test the SVILP method shows significant improvement with respect to the other methods. We finally compared SVILP with TOPKAT. The software accepts the structures of the molecules in SMILES string and automatically split the molecule into different fragments, and then uses these fragments as well as some chemical descriptors such as LOGP and shape index for predictions. In order to make

	MSE	R-squared
CHEM	1.04	0.48
PLS	1.03	0.47
TOPKAT	2.2	0.26
SVILP	**0.8**	**0.57**

Fig. 10. MSE and R-squared for CHEM, PLS, TOPKAT and SVILP

a fair comparison of the above methods with the commercial software TOPKAT, we must ensure that we only consider predicted accuracies for molecules that were not included in the training data of either method. We therefore excluded any of the DSSTox molecules that TOPKAT had in its database leaving 165 unseen molecules. Figure 10 shows the results. According to sign test, the SVILP shows significant improvement in comparison with all of the other approaches. Our results show that SVILP outperforms all the other methods in the study. The results confirm the efficacy and usefulness of our approach.

6 Conclusions and Further Work

In this paper we introduce a new framework for combining Support Vector machine technology with Inductive Logic Programming. Unlike existing relational kernels, the present approach works within the standard ILP setting of generalisation with respect to background knowledge, rather than the limited setting of atomic generalisation. A particular kernel is defined and implemented on top of the ILP system CProgol5.0. This kernel has been tested on an important new toxin dataset. In our experiments we compared the performance of the SVILP against related approaches. In all cases our approach produced significantly higher predictive accuracy.

Further theoretical work is necessary to clarify the effects on performance of varying the amount of background knowledge used by the kernel. Also further empirical work is needed to test the kernel on a wider variety of relational problems.

Acknowledgements

The authors would like to acknowledge the support of the DTI Beacon project "Metalog - Integrated Machine Learning of Metabolic Networks Applied to Predictive Toxicology", Grant Reference QCBB/C/012/00003 and the ESPRIT IST project "Application of Probabilistic Inductive Logic Programming II (APRIL II)", GrantRef: FP-508861.

References

1. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: Proceedings of the Nineteenth International Conference on Machine Learning. (2002) 176–186
2. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels for structured data. In Matwin, S., Sammut, C., eds.: Proceedings of the Twelfth International Conference on Inductive Logic Programming. LNAI 2583, Berlin, Springer-Verlag (2002) 66–83
3. Plotkin, G.: A note on inductive generalisation. In Meltzer, B., Michie, D., eds.: Machine Intelligence 5. Edinburgh University Press, Edinburgh (1969) 153–163
4. Plotkin, G.: Automatic Methods of Inductive Inference. PhD thesis, Edinburgh University (1971)
5. Page, D., Frisch, A.: Generalization and learnability: A study of constrained atoms. In Muggleton, S., ed.: Inductive Logic Programming. Academic Press, London (1992)
6. Lloyd, J.: Logic for Learning. Springer, Berlin (2003)
7. Chevaleyre, Y., Zucker, J.: A framework for learning rules from multiple instance data. In: Proceedings of the European Conference on Machine Learning (ECML 2001), Berlin, Springer-Verlag (2001) 49–60 LNAI 2167.

8. Dumais, S., Platt, J., Heckermann, D., Sahami, M.: Inductive learning algorithms and representations for text categorisation. In: Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management. (1998) 148–155
9. Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag, New York (1995)
10. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society London (A) **209** (1909) 415–446
11. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California in Santa Cruz, Computer Science Department (1999)
12. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research **2** (2002) 419–444
13. Horváth, T., Gaertner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2004) 158–167
14. Muggleton, S.: Inductive Logic Programming. New Generation Computing **8** (1991) 295–318
15. King, R., Whelan, K., Jones, F., Reiser, P., Bryant, C., Muggleton, S., Kell, D., Oliver, S.: Functional genomic hypothesis generation and experimentation by a robot scientist. Nature **427** (2004) 247–252
16. Sternberg, M., Muggleton, S.: Structure activity relationships (SAR) and pharmacophore discovery using inductive logic programming (ILP). QSAR and Combinatorial Science **22** (2003)
17. Muggleton, S.: Bayesian Inductive Logic Programming. In Warmuth, M., ed.: Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, New York, ACM Press (1994) 3–11 Keynote presentation.
18. Muggleton, S.: Inverse entailment and Progol. New Generation Computing **13** (1995) 245–286
19. Kramer, S., Lavrac, N., Flach, P.: Propositionalisation approaches to Relational Data Mining. In Dzeroski, S., Larac, N., eds.: Relational Data Mining. Springer, Berlin (2001) 262–291
20. Lavrač, N., Džeroski, S., Grobelnik, M.: Learning non-recursive definitions of relations with LINUS. In Kodratoff, Y., ed.: Proceedings of the 5th European Working Session on Learning, volume 482 of Lecture Notes in Artificial Intelligence, Springer-Verlag (1991)
21. Kramer, S., Pfahringer, B., Helma, C.: Stochastic propositionalisation of non-determinate background knowledge. In: Proceedings of the Eighth International Conference on Inductive Logic Programming, Berlin, Springer-Verlag (1998) 80–94
22. Srinivasan, A., King, R.: Feature construction with inductive logic programming: a study of quantitative predictions of biological activity aided by structural attributes. Data Mining and Knowledge Discovery **3** (1999) 35–57
23. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Mining and Knowledge Discovery **3** (1999) 7–36
24. Kramer, S., E, F.: Bottom-up propositionalisation. In: Proceedings of the ILP-2000 Work-In-Progress Track. Imperial College, London (2000) 156–162
25. Mavroeidis, D., Flach, P.: Improved distances for structured data. In Horváth, T., Yamamoto, A., eds.: Proceedings of the Thirteenth International Conference on Inductive Logic Programming. LNAI 2835, Berlin, Springer-Verlag (2003) 251–268
26. Cumby, C., Roth, D.: On kernel methods for relational learning. In: Proceedings of the Twentieth International Conference on Machine Learning. (2003) 107–114
27. Gaertner, T., Driessens, K., Ramon, J.: Graph kernels and gaussian processes for relational reinforement learning. In: Proc. of the 13th International Conference on Inductive Logic Programming, Springer Verlag (2003) 146–163

28. Ramon, J., Bruynooghe, M.: A framework for defining distances between first-order logic objects. In Page, D., ed.: Proceedings of the Eighth International Workshop on Inductive Logic Programming (ILP98), Berlin, Springer-Verlag (1998) 271–280 LNAI 1446.
29. Horvath, T., Wrobel, S., Bohnebeck, U.: Relational instance-based learning with lists and terms. Machine Learning **43** (2001) 53–80
30. Nienhuys-Cheng, S.: Distance between Herbrand interpretations: a measure for approximations to a target concept. In Lavrač, N., Džeroski, S., eds.: Proceedings of the Seventh International Workshop on Inductive Logic Programming (ILP97), Berlin, Springer-Verlag (1997) 321–226 LNAI 1297.
31. King, R., Muggleton, S., Srinivasan, A., Sternberg, M.: Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. Proceedings of the National Academy of Sciences **93** (1996) 438–442
32. Richard, A., Williams, C.: Distributed structure-searchable toxicity (DSSTox) public database network: A proposal. Mutation Research **499** (2000) 27–52
33. Pearlman, R.S.: Concord User's Manual. Tripos, Inc, St Louis, Missouri (2000)
34. Collobert, R., Bengio, S.: Svmtorch: Support vector machines for large-scale regression problems. Journal of Machine Learning Research **1** (2001) 143–160