Pertinent Background Knowledge for Learning Protein Grammars

Christopher H. Bryant[1,⋆], Daniel C. Fredouille[1], Alex Wilson[2], Channa K. Jayawickreme[3], Steven Jupe[4], and Simon Topp[5]

[1] School of Computing, The Robert Gordon University, Aberdeen UK
chb@comp.rgu.ac.uk
[2] School of Computing, Division of Mathematics and Statistics, The Robert Gordon University, Aberdeen UK
[3] Discovery Research Biology, GlaxoSmithKline, Durham, USA
[4] Department of Bioinformatics, GlaxoSmithKline, Stevenage, UK
[5] Department of Bioinformatics, GlaxoSmithKline, Harlow, UK

Abstract. We are interested in using Inductive Logic Programming (ILP) to infer grammars representing sets of protein sequences. ILP takes as input both examples and background knowledge predicates. This work is a first step in optimising the choice of background knowledge predicates for predicting the function of proteins. We propose methods to obtain different sets of background knowledge. We then study the impact of these sets on inference results through a hard protein function inference task: the prediction of the coupling preference of GPCR proteins. All but one of the proposed sets of background knowledge are statistically shown to have positive impacts on the predictive power of inferred rules, either directly or through interactions with other sets. In addition, this work provides further confirmation, after the work of *Muggleton et al., 2001* that ILP can help to predict protein functions.

1 Introduction

Inductive Logic Programming (ILP) has tackled many molecular biological applications such as: secondary structure prediction [1, 2], Mutagenic activity of small molecules [3], prediction of genes' functions [4, 5] and prediction of functions of proteins [6]. We are interested in this last application, where grammars inferred from protein sequences have been shown, through a case study, to help to predict the function of proteins. This paper provides a comparison of sources of background knowledge that can be used in such tasks. It is also a confirmation, with stronger statistical evidence than [6], of the utility of ILP inferred rules in predicting protein functions.

The next section introduces protein grammar inference via ILP. Section 2 presents different sets of background knowledge predicates that can be used for inference of grammars over proteins. Section 3 evaluates the main effects and interactions of the different sets using a reliability engineering method known as *Taguchi design* [7], 10-folds cross-validations are used to draw the final conclusions.

⋆ Contact author.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 54–65, 2006.

Protein Grammar Inference with ILP

Patterns in the form of grammars have been used with success to model protein families. The use of such grammars is twofold: (1) they can be used to annotate sequences of unknown function, providing molecular biologists with a likely function for such sequences; (2) they can help biologists to understand how biological functions are realised because the grammar structure represents common points between sequences of similar functions. Many grammar formalisms have been used, including String Variable Grammars (SVG) [8], Patscan patterns [9], Prosite patterns [10, 11], Basic Gene Grammars (BGG) [12] and Probabilistic Regular or Context-Free Grammars [13, 14]. The hand development of grammars, using for example SVG or BGG formalisms, is difficult and requires expensive human expertise. Moreover, some patterns might be too subtle to be recognised by a human expert. Thus, given the enormous volume of data arising from genome projects, the acquisition of grammars from sets of biological sequences needs to be automated.

ILP has two advantages in this application domain: first ILP infers logic programs, and logic programs have been shown useful to represent hand designed protein grammars (*e.g.*, with SVG [8]); second, unlike most machine learning techniques, ILP is able to bias inference to take expert knowledge into account. This is certainly an advantage in this application domain since, as protein sequences are not just sequences but represent molecules with physical and chemical properties, expert knowledge is often available. However, as providing more background knowledge predicates enlarges the search space, a compromise between the space size and the amount of knowledge introduced has to be found.

Different approaches to grammar learning with ILP have been considered, mainly by Cussens and Pulman [15] and Muggleton *et al.* [6]. These papers differ in two main points. First the application in [15] is natural language while the one in [6] is molecular biology. Second, the logic representation in [15] uses chart parsing tables, while Definite Clause Grammars (DCG) [16] are used in [6]. Our inference approach takes its roots in the work of Muggleton *et al.* [6] and can be summarised as follows. The inference process takes as inputs: (1) examples (and counter-examples if available) of the form `target(L,[]).` where `L` is a list representing the primary structure of the example protein, *i.e.*, the sequence of its amino-acids (*e.g.*, `L=[n,n,e,v,...]`) and `[]` is a list which is empty *i.e.*, has no elements; and (2) background knowledge predicates of the form $\texttt{pred}_i\texttt{(+IL,-OL).}$ where `IL` is the input list of amino-acids, and `OL` is the output list, which is a suffix of `IL`, obtained by removing the amino-acids matched by the predicate from `IL`. From these, the inference process infers rules of the form $\texttt{target(A,B):- pred}_1\texttt{(A,C), pred}_2\texttt{(C,D),...pred}_n\texttt{(X,B).}$ which maximises the score function of the ILP system. For further details see [6].

This study proposes sets of background knowledge predicates for protein grammar learning (*i.e.*, the $\texttt{pred}_i$), and a statistical study of the influence of these sets on the predictive power of inferred rules.

BKS	Example predicate	Rules
$\mathcal{L}et$	a/2	a([a|B],B).
$\mathcal{P}ro$	tiny/2	tiny([a|B],B). tiny([g|B],B). tiny([s|B],B)
$\mathcal{G}_u$	gap/2	gap(A,A). gap([_|A],B):-gap(A,B).
$\mathcal{G}_s$	x0_1/2	x0_1(A,A). x0_1([_|A],A).
$\mathcal{S}_p$ or $\mathcal{S}_n$	dry/2	dry([d,r,y|B],B).
$\mathcal{P}_a$	pratt1/2	pratt1(A,B):- h(A,C), t_or_i(C,D), x0_1(D,E), tiny(E,F), t(F,B). t_or_i([t|A],A). t_or_i([i|A],A).
$\mathcal{P}_s$	pratt_sub1/2	pratt_sub1(A,B):- h(A,C), x0_1(C,D), t_or_i(D,E).

Fig. 1. Examples for the different BKSs studied in this paper

2 Protein Sequence Background Knowledge

We can split the background knowledge into two main categories: general molecular biology knowledge (Subsection 2.1), and knowledge specific to each particular data-set (Subsection 2.2). In the following, a set of background knowledge predicates obtained by a common procedure is denoted by BKS (Background Knowledge Set).

2.1 General Molecular Biology Knowledge

This subsection considers expert knowledge which can *a-priori* be considered relevant for any protein grammar inference process. We can split such knowledge in two parts: (1) amino-acid letters and their physico-chemical properties, (2) gaps. Except for some gaps predicates, these predicates have already been used to infer biological grammars in [6].

Amino-Acid Letters and Properties. The two first BKSs we consider are predicates matching exactly one amino-acid letter (denoted by $\mathcal{L}et$), and predicates matching sets of amino-acid with common physico-chemical properties (denoted by $\mathcal{P}ro$). The use of these BKSs is motivated by the knowledge that the conservation – of amino-acids for $\mathcal{L}et$, or of physico-chemical properties for $\mathcal{P}ro$ – at some positions in the proteins can often help predicting the protein function. Different physico-chemical properties can be considered; for this work, we used those proposed by [1] and also used in [6]. Examples of predicates for the $\mathcal{L}et$ and $\mathcal{P}ro$ BKSs are given in Figure 1.

Gaps. Protein sequences contain parts participating to the overall structure of the molecule but which are either not directly relevant to the function or which cannot be characterised by the provided background predicates. To match such parts of the protein, we can use predicates called *gaps*; we consider two types of gaps: *unlimited* and *short* gaps. An unlimited gap is a predicate which can match any sequence. We will denote this BKS by $\mathcal{G}_u$. There is just one predicate for $\mathcal{G}_u$, namely `gap/2` (see Figure 1). The second BKS, short gaps, denoted by

$\mathcal{G}_s$, contains predicates matching sequences with small length (we considered predicates matching sequences with lengths from 0 to 1, 0 to 2, 1 to 1, 1 to 2, and 2 to 2). As an example the predicate matching sequences with lengths from 0 to 1 is given in Figure 1. While unlimited gaps can cover large uncharacterised parts of proteins, short gaps can help the discovery of well conserved groups of amino-acids separated by a few, less conserved, amino-acids. Some biological grammars contains gaps matching a range of large lengths. In this work, we considered that they can be approximated by the `gap/2` predicate.

2.2 Sequence Family Knowledge

In addition to the generic BKSs discussed above, BKSs on the particular protein family under study are available. These BKSs can be obtained from two sources: from experts on that protein family, or by automatically processing the examples. Since the availability and quality of background knowledge provided by experts can vary, it is not taken into account in this study. We therefore focus on knowledge that can be automatically extracted from the training examples, before inference.

Subsequences. We consider providing *exceptionally frequent* subsequences of the positive examples to the ILP system. We proceed in four steps. During step (1), we extract subsequences that are present in at least 10% of the positive training set. This enables inferred rules using the subsequences to cover a reasonable amount of examples. Let $Obs(s)$ be the number of positive examples containing subsequence s. During step (2), we define a distribution over subsequences and compute for each subsequence s the number of times, denoted $Exp(s)$, s is expected to appear in the examples. We consider two distributions detailed in the paragraphs below. In step (3), we score each subsequence using the value $\frac{(Obs(s)-Exp(s))^2}{Exp(s)}$. This score function was proposed in [17] for the extraction of exceptional subsequences in biological sequences. In step (4) the subsequences are ranked using the score and only the best 40 are kept. The next two paragraphs detail the two distributions used for step (2).

Distribution over the positive examples. Using a distribution based on the positive examples enables to detect subsequences describing the positive examples. To obtain such a distribution, we use VERBUMCULUS [17]. VERBUMCULUS trains a Markov Model (MM) on the provided sequences. The expected frequency of the subsequences with respect to the MM distribution can be extracted from VERBUMCULUS output. The order of the MM is an important parameter: when an order of O is taken, only subsequences longer than $O + 1$ have frequency which can be different from the MM expected frequency. To use the maximum amount of information available in the positive examples, we therefore trained a MM of order $L-2$ to obtain the expected frequencies for subsequences of length L. The subsequences obtained using this distribution are denoted by $\mathcal{S}_p$.

Distribution over the negative examples. Subsequences generated from the above distribution may be present in the negatives as well as in the positives. Subsequences discriminating between positive and negative examples can be obtained by using a distribution based on the set of negative examples. In this case, the distribution and the extracted subsequences are not obtained from the same set of sequences. In consequence we can use a simpler method than the one proposed for the previous distribution. Instead of using a MM, the expected frequency of a subsequence is estimated by $Exp = count(sub) * \frac{P}{N}$, where $count(sub)$ is the number of negative examples containing the subsequence, and P and N are respectively the positive and negative training set size. The subsequences obtained using this distribution are denoted by $\mathcal{S}_n$.

Pratt. In addition to subsequence extraction, software already exist to extract common points in protein sequences and represent them as patterns. The most popular is certainly PRATT [10]. The patterns inferred by PRATT are obtained using only positive examples. PRATT patterns can be seen as simplified regular expressions, an example of such a pattern is: `H-[TI]-x(0,1)-[KRH]-T`. An equivalent DCG is provided in Figure 1, line $\mathcal{P}_a$. Such patterns are widely used by molecular biologists, as shown by their availability in the Prosite database [11]. We propose to use these patterns as they stand (BKS denoted by $\mathcal{P}_a$) or to extract smaller patterns from them (denoted by $\mathcal{P}_s$). For $\mathcal{P}_s$, we extracted all sub-patterns containing two non gap elements. For example, sub-patterns `H-[TI]`, `[TI]-x(0,1)-[KHR]` and `[KHR]-T` can be extracted from the above pattern (see also Figure 1, line $\mathcal{P}_s$). This second usage aims at compensating for the fact that PRATT cannot take into account counter-examples: it potentially returns patterns frequent in both the positive and negative examples sets. Refinements of $\mathcal{P}_s$ by the ILP system could help the creation of patterns rejecting the negatives.

3 Evaluation of Background Knowledge Effects

Subsection 3.1 presents the inference task. Subsection 3.2 explains the experiments that evaluate the influence of the different BKSs on inference and subsection 3.3 discusses the experimental results. Experimental materials are available at: `http://www.comp.rgu.ac.uk/staff/chb/research/data_sets/ecml06/bk` .

3.1 Description of the Inference Data and Task

Data-set. G-protein coupled receptors (GPCRs) are the biggest single class of receptors in biology. An understanding of how they couple with specific classes of G-proteins is vital for further comprehending the function of the receptor within a cell. The data set consists of two sets of sequences representing two qualitatively distinct classes, Gi/o and Gs/q, of GPCRs [18]. Gi/o and Gs/q are

the *coupling specificity* of the GPCRs proteins. Data allowing the classification of these proteins into the two sets is proprietary to GlaxoSmithKline (GSK), the industrial collaborator of this project. The Gi/o and Gs/q data-sets contain 64 and 126 sequences respectively. The task we consider is to infer rules which classify GPCRs as either Gi/o or Gs/q. It is possible that some GPCRs have both the Gi/o and Gs/q properties, however the sequences in our data-set are known to belong only to one of these classes.

Different papers tackle the prediction of GPCR coupling using machine learning. These include the use of regular expressions, Naïve Bayes and Hidden Markov Models (see [19] for a good overview). The state of the art methods providing the best classifications are very specialised to the GPCR coupling prediction task [19], showing the difficulty of the task. Our aim in this paper is not to provide a better classifier than the existing ones, but to evaluate the effect of the BKSs on this hard inference task, using generic ILP methods.

GPCRs have a characteristic 7 membrane-spanning regions and thus have regions outside the cell, within the cell membrane and inside the cell. It is believed that the G-protein binding property depends only on the subsequences of GPCRs situated inside the cell. We therefore only considered these four intracellular subparts during the inference processes. This means that the original data-set can be separated into eight sets, four containing Gi/o sequences and four containing Gs/q sequences; the four data-sets associated to each class corresponding to the four intra-cellular subparts. The length of these subsequences varies from 11 to 23 in the first subpart, from 16 to 42 in the second, 21 to 245 in the third, and finally 21 to 172 in the fourth. Because the limits of subsequences in each subpart are not always well defined, we decided to infer patterns conserved *inside* the subparts, therefore all bodies of inferred rules for this work start and end with the `gap/2` predicate. Providing (or not) $\mathcal{G}_u$ to the inference process therefore means that we allow (or not) the `gap/2` predicate to be present somewhere other than at the beginning or end of inferred rules' body.

We created cross-validation sets from this data. Our method of partitioning the data ensured training and test sets never contained homologous sequences. To ensure this: (1) we concatenated the four intra-cellular subparts of each GPCR; (2) we created clusters over these sequences with BlastClust [20] (these clusters are based on homology between the sequences); and finally (3), clusters (instead of sequences) are randomly put in n-disjoint sets which are then used to create a n-folds cross-validation set (we used n=5 and n=10, see Subsection 3.2).

Predictions. To be able to make predictions, we have to combine the 8 inferred sets of rules which are obtained by: (1) inferring on the four different intra-cellular subparts, (2) using either Gi/o or Gs/q as positive examples (the examples of the other class being used as negatives). For a given rule r, let $pr(r)$ be its precision over the Gi/o training examples, *i.e.*, $pr(r) = \frac{p}{p+n}$ where p (resp. n) is the number of training Gi/o examples (resp. Gs/q examples) accepted by r. For each sequence to classify, we parse each of its intra-cellular subparts with

the associated inferred rules[1]. Let R be the set of rules matching the sequence (on the respective subparts they have been inferred on). The sequence is then associated with the average obtained precisions over the matching rules, *i.e.*, the value $\sum_{r \in R} \frac{pr(r)}{|R|}$. The larger the obtained value, the more likely the sequence is Gi/o, the smaller, the more likely the sequence is Gs/q. This strategy has been used mainly because it is a simple way to weight rules using information from their training set performance.

ILP System and Parameters. We used the ALEPH ILP inference platform (`http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph`) to run our experiments. our previous work which provides large speed-ups of ALEPH for biological grammar inference.

The Gi/o and Gs/q sets contain a very different number of sequences while having the same importance to the biologists. Therefore, to avoid biasing the inference toward one class, we decided to weight, in the ILP system evaluation function, the examples of each class by the inverse of the number of instances of the class available. The evaluation function used is the accuracy over the weighted examples, *i.e.*, $\text{acc} = \frac{1}{2} * (\frac{p}{P} + \frac{n}{N})$, where P (resp. N) is the size of the positive (resp. negative) training set size, and p (resp. n) is the number of positive (resp. negative) training examples covered (resp. rejected) by the rule.

To prevent over-fitting, we consider that a rule is valid only if it covers at least 10% of the positives training examples. To prevent the inference of over-general rules, we constrain the inferred rules to accept a proportion of the positives larger than 1.5 times the proportion of accepted negative (*e.g.*, rules like `target(A,B):-gap(A,B).` are rejected thanks to this condition). Finally, based on results of preliminary experiments, we limited the explored part of the space by setting the parameters `nodes` to 50000 and `depth` to 7.

3.2 Design of Experiments

We designed experiments to answer the following questions: (Q_1) Which combinations of BKSs improve the results of the ILP inference processes, and which make it worse? (Q_2) Does the expected best combination of BKSs actually result in a predictor with significantly high predictive power?

To be able to answer (Q1), we have to sample the space of combinations of BKSs. We have eight different background knowledge types that we want to test ($\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{L}et$, $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{P}ro$, $\mathcal{S}_n$, $\mathcal{S}_p$), which can be combined in $2^8 = 256$ different ways. We could not try all combinations because the running times are too long. We therefore had to sample the space of combinations. This was done using the technique known as *Taguchi design* [7]. The Taguchi method takes care to select a set of samples balanced with respect to the use of the different factors (here the BKSs), and of selected sets of interactions between

[1] To ensure there is a score for each of the eight sets of rules, each set is completed with a "default" rule, used when no other matches. The precision of the default rule is set as the number of rejected (by all other rules) Gi/o training examples over the number of rejected Gi/o and Gs/q examples.

the factors (*i.e.*, the effects of combining different BKSs). The effects of these factors and interactions can then be studied independently without sampling biases. We selected our Taguchi design among those available in the statistical software MINITAB (`http://www.minitab.com/`), taking the one allowing for the study of the largest number of interactions. This design requires 32 samples, (*i.e.*, inference over 32 different combinations of BKSs) and allows for the study of 20 interactions between two BKSs. Because gaps by themselves cannot produce interesting rules and are expected to interact, we chose to study interactions between $\mathcal{G}_u$ and all other BKSs, and $\mathcal{G}_s$ and all other BKSs. Other available interactions were fixed by MINITAB and are between $\mathcal{L}et$ and all other BKSs, between $\mathcal{P}_a$ and $\mathcal{S}_p$, and between $\mathcal{P}_a$ and $\mathcal{S}_n$.

To augment the statistical significance of our results, we used a 5-folds cross-validation: the number of the fold being considered as a noise parameter in the Taguchi design. We limited ourselves to a 5-folds due to execution time constraints: one combination of BKSs in a 5-folds experiment takes approximately 60 hours to run on a *SunBlade 2500* processor under *Sun0S 5.8*. Hence a total of $60 * 32 = 1920$ hours of cpu time. ROC area on the cross-validation test sets has been chosen as a predictive performance measure. One of its main advantages is that it is independent of the proportions of classes in the test sets.

The analysis of the Taguchi experiments was used to answer question (Q_1). This analysis was conducted by examining Taguchi graphs (available on the web), and fitting the responses (*i.e.*, ROC areas) to a linear model of the different factors (the BKSs), and available interactions. The coefficients of the linear model provide indications of the amount of ROC area each BKS (or interaction) brings or remove to the total area. These coefficients are associated with p-values which represent their significance, *i.e.*, the estimated probability that the hypothesis "the effect is nul" is true. A value of 0.05 (*i.e.*, 5%) is a usual significance threshold. Finally, a R^2 value is provided, representing the percentage of the ROC area variation explained by the linear model.

Using the linear model, we can predict which combinations of BKSs will improve the ROC area. The best combinations were tested by using 10-folds cross-validation, enabling a final selection of the BKSs. (Q_2) was then answered by comparing the ROC area for this selection with random classification.

3.3 Experimental Results and Analysis

This section presents the statistical analysis of the results; full tables of results are available on the web. For the analysis, we first constructed a linear model with all available terms (*i.e.*, main effects and interactions). Then, assuming that effects with large p-values are random, as per Taguchi strategy in small design, interaction terms with large p-values (over 0.4) have been removed and a new model was constructed. The obtained model is given in Table 1. The lower bound estimation for the R^2 value of this model is 82.9%, *i.e.*, approximately 17% of the variation has to be explained by parameters not included in the model (*e.g.*, other unavailable second order interactions, or higher order interactions).

Table 1. Linear model of the ROC areas. P-values are expressed as percentages.

Main effects	Coef.	p-value	Interactions	Coef.	p-value
Constant	54.75	0.0	$\mathcal{L}et$-$\mathcal{P}ro$	-0.81	4.1
$\mathcal{L}et$	0.27	46.4	$\mathcal{L}et$-$\mathcal{S}_p$	-0.89	2.7
$\mathcal{P}ro$	2.49	0.0	$\mathcal{G}_s$-$\mathcal{L}et$	1.16	0.6
$\mathcal{G}_s$	2.59	0.0	$\mathcal{G}_s$-$\mathcal{P}_s$	0.92	2.2
$\mathcal{G}_u$	1.51	1.0	$\mathcal{G}_u$-$\mathcal{S}_n$	0.78	4.7
$\mathcal{P}_a$	0.02	95.2	$\mathcal{G}_u$-$\mathcal{S}_p$	0.70	7.3
$\mathcal{P}_s$	0.59	12.3	$\mathcal{P}_a$-$\mathcal{S}_n$	1.04	1.1
$\mathcal{S}_p$	0.39	30.2			
$\mathcal{S}_n$	0.55	14.8			

Table 2. Results for combinations of BKSs suggested by the linear model (4 first lines), and some complementary experiments (last 3 lines). The "Pred." column corresponds to predictions of mean ROC areas by the linear model on 5-folds experiments.

	Combination	5-folds			10-folds	
		Mean	Med.	Pred.	Mean	Med.
BKS_1	$\mathcal{G}_s$-$\mathcal{G}_u$-$\boldsymbol{\mathcal{L}et}$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\boldsymbol{\mathcal{S}_p}$-$\mathcal{S}_n$	60.5	63.4	66.1	71.6	75.3
BKS_2	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\boldsymbol{\mathcal{S}_p}$-$\mathcal{S}_n$	61.7	62.5	66.6	61.3	64.6
BKS_3	$\mathcal{G}_s$-$\mathcal{G}_u$-$\boldsymbol{\mathcal{L}et}$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_n$	60.6	61.8	65.7	71.4	75.0
BKS_4	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_n$	61.2	61.3	62.7	57.9	66.7
BKS_5	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_p$-$\mathcal{S}_n$	61.0	67.1	63.9	69.8	74.6
BKS_6	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_p$	60.7	61.8	61.3	70.3	73.1
BKS_7	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_a$-$\mathcal{P}ro$-$\mathcal{S}_p$-$\mathcal{S}_n$	65.3	65.9	63.0	59.1	60.4

Usefulness of $\mathcal{G}_u$, $\mathcal{G}_s$, $\mathcal{P}ro$, $\mathcal{P}_a$, $\mathcal{P}_s$ and $\mathcal{S}_n$. In the linear model $\mathcal{G}_u$, $\mathcal{G}_s$ and $\mathcal{P}ro$ have the largest main effects coefficients, significant at the 5% level. In addition, $\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{P}_a$, $\mathcal{P}_s$, and $\mathcal{S}_n$ are shown to have positive interactions at the 5% significance level. This is strong evidence that these BKSs have to be used.

Interactions with gaps ($\mathcal{G}_s$ and $\mathcal{G}_u$) were expected: gaps cannot describe the sequences by themselves, and in fact the importance of their main effects can be seen as an indication that they positively interact with others background predicates most of the time.

The effects of $\mathcal{L}et$ and $\mathcal{S}_p$ are less clear than for the others BKSs. For $\mathcal{L}et$, two negative interactions are observed (with $\mathcal{P}ro$ and $\mathcal{S}_p$), and one positive with $\mathcal{G}_s$. For $\mathcal{S}_p$, in addition to the negative interaction with $\mathcal{L}et$, a positive interaction is observed with $\mathcal{G}_u$; this interaction is observed at a 10% significance level instead of a stronger 5% level for the negative interaction. Further experiments are therefore needed to prove utility of the $\mathcal{L}et$ and $\mathcal{S}_p$ BKSs.

Usefulness of $\mathcal{L}et$ and $\mathcal{S}_p$. The linear model suggests different combinations to test: always using $\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{P}ro$ and $\mathcal{S}_n$, but adding or not $\mathcal{L}et$ and $\mathcal{S}_p$ (or both). Results for these 4 combinations both with 5-folds experiments (used for the Taguchi design) and 10-folds ones are in lines BKS_1 to BKS_4 in Table 2.

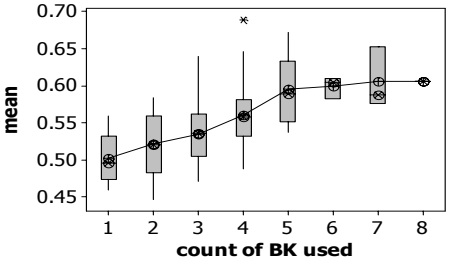

Fig. 2. Mean ROC areas as a function of the number of BKSs used. The plot contains points obtained for the Taguchi design, but also extra-points obtained during preliminary experiments.

The $\mathcal{L}et$ BKS: Wilcoxon Signed Rank tests have been used to test differences between the results of BKS_1 and BKS_2, and of BKS_3 and BKS_4[2] (*i.e.*, comparing inferences with and without $\mathcal{L}et$). On the 5-folds data, no evidence of differences in the results medians is available (at the 10% significance level). However, on the 10-folds data, their is evidence at the 5% significance level that BKS_2 has lower median than BKS_1; and at the 1% significance level that BKS_4 has lower median than BKS_3. The $\mathcal{L}et$ BKS can therefore be considered useful with strong evidence.

The $\mathcal{S}_p$ BKS: Wilcoxon Signed Rank tests between BKS_1 and BKS_3, and BKS_2 and BKS_4 (*i.e.*, comparing inferences with and without $\mathcal{S}_p$) do not detect differences, at the 10% significance level, between these results medians. Therefore, we do not have any statistical evidence that providing $\mathcal{S}_p$ changes inference results.

Is Classification Better Than Random? From the previous results, the best obtained combinations are BKS_1, followed very closely by BKS_3. Using a 1-Sample Wilcoxon test, it can be shown that, for both these combinations, the median is above random (*i.e.*, a value of 50.0), at the 10% significance level on the 5-folds, and at a strong 1% significance level on the 10-folds data. This confirms, after the work of [6], that protein grammars inferred by ILP can be useful for predicting protein functions: a stronger statistical evidence is provided in this work thanks to 10-folds cross-validation (holdout was used in [6]).

High order interactions. Each provided background predicate enlarges the search space, we therefore could expect performance to decrease when adding many BKSs. Two facts tend to show the effect of the search space size can be observed. First, the predictions from the linear model (column *Pred.* in Table 2) are most of the times lower than the results of the practical experiments. Second, Figure 2 shows a plot of mean ROC areas with respect to the number of BKSs used; on this figure, improvements in ROC areas are smaller when more than 5

[2] The Wilcoxon Signed Rank test is used instead of a more classical paired t-test since the differences of distributions cannot be assumed to be normal.

BKSs are provided. Both observations could be explained by the presence of a negative high order interaction like the search space size effect.

If such an interaction takes place, and that BKSs not studied in this work are considered for inference, using them in addition to proposed ones could lower the results. If this is observed, a solution would be to replace BKSs of this study having low contribution by the new one(s). The first BKS suggested for replacement, both by the linear model and by the 10-folds experiments, is $\mathcal{S}_p$. If this does not prove enough, Table 2 suggests removing $\mathcal{P}_a$ or $\mathcal{S}_n$.

Sensitivity to the Examples Count. Different ROC areas are often observed between the 5-folds and 10-folds results (*e.g.*, for BKS_1 in Table 2, but more can be seen on the data table available on the web). This may be due to a sensitivity of the ILP system to the size of the training set available to inference. It may also be due to higher quality BKSs being generated when more sequences are available.

Processing the BKSs. To obtain more insight on generated BKSs of Section 2.2, we ran 10-folds experiments using all BKSs except either $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{S}_n$ or $\mathcal{S}_p$ (Table 2). When using a Wilcoxon Signed Rank test to compare these results with BKS_1, the medians of the results are not shown different at the 10% significance level. However, the Wilcoxon Signed Rank test with the lowest p-value is with BKS_7, *i.e.*, when the $\mathcal{P}_s$ BKS is removed (p-value of 11.8%)[3]. $\mathcal{P}_s$ is also the generated BKS with the lowest p-value (12.3%) on the linear model. This makes us believe that $\mathcal{P}_s$ is likely to be the best generated BKS.

$\mathcal{P}_s$ is the only generated BKS which was not obtained directly from the examples, but obtained by processing another BKS. This encourages us to think that re-working the BKSs obtained from the examples is a possible way to improve further the inference results.

4 Conclusion

This work provides statistical evidence that all but one of the proposed BKSs are useful to inference, sometimes directly, sometimes through interactions with each other. It also provides further confirmation, after the work of Muggleton *et al.* [6] that ILP can help to predict protein functions.

Other sources of background knowledge have still to be studied, these include known regular expressions (*e.g.*, from the Prosite database [11]), but also probabilistic grammars (*e.g.*, weight matrices or Markov models).

Acknowledgements. This work is funded by EPSRC grant GR/S68682. We would like to acknowledge the contributions made by the Systems Research, Transgenics & Gene Cloning, and Gene Expression & Protein Biochemistry groups at GlaxoSmithKline to the proprietary GPCR classification data.

[3] This p-value is not smaller despite the large difference in median of BKS_1 and BKS_7 because these combinations do not perform well on the same folds.

References

1. Muggleton, S., *et al.*: Protein secondary structure prediction using logic-based machine learning. Protein Eng. **5** (1992) 647–657
2. Mozetic, I.: Secondary structure prediction by inductive logic programming. In: Proc. 3rd Meeting on the Critical Assessment of Techniques for Protein Structure Prediction, CASP3. (1998) pp. A–26
3. Srinivasan, A., *et al.*: Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence **85**(1-2) (1996) 277–299
4. King, R.D.: Applying inductive logic programming to predicting gene function. AI Mag. **25**(1) (2004) 57–68
5. Clare, A., *et al.*: The ILP'05 challenge. `http://www.protein-logic.com/index.html` (2005)
6. Muggleton, S.H., *et al.*: Are grammatical representations useful for learning from biological sequence data? – a case study. Jour. Comp. Biol. **5**(8) (2001) 493–522
7. Taguchi, G.: Introduction to quality engineering. Asian Productivity Organization, Tokyo (distributed by American Supplier Institute, Inc., Dearborn,MI.) (1986)
8. Searls, D.B.: String variable grammar: A logic grammar formalism for the biological language of DNA. Jour. of Log. Prog. **12** (1993)
9. Dsouza, M., *et al.*: Searching for patterns in genomic data. Trends in Genetics **13**(12) (1997) 497–498
10. Brazma, A., *et al.*: Discovering patterns and subfamilies in biosequences. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, AAAI Press (1996) 34–43
11. Falquet, L., *et al.*: Protein data bank. Nucleic Acid Research **30** (2002) 235–238
12. Leung, S.W., *et al.*: Basic Gene Grammars and DNA-ChartParser for language processing of *Escherichia coli* promoter DNA sequences. Bioinformatics **17**(3) (2001) 226–236
13. Bateman, A., *et al.*: The Pfam protein families database. Nucleic Acids Research **32** (2004) D138–D141
14. Sakakibara, Y., *et al.*: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research **22** (1994) 5112–5120
15. Cussens, J., Pulman, S.: Experiments in inductive chart parsing. In Cussens, J., ed.: LLL'99, Bled, Slovenia (1999) 72–83
16. Pereira, F., Warren, D.H.D.: Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence **13**(3) (1980) 231–278
17. Apostolico, A., *et al.*: Verbumculus and the discovery of unusual words. Jour. Comp. Sci. and Tech. **19**(1) (2003) 22–41
18. Pierce, K., *et al.*: Seven-transmembrane receptors. Nat Rev Mol Cell Biol **3(9)**(6) (2002) 39–50
19. Sgourakis, N., *et al.*: Prediction of the coupling specificity of GPCRs to four families of G-proteins using hidden markov models and artificial neural networks. Bioinformatics **21(22)** (2005) 4101–4106
20. Altschul, S., *et al.*: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Res. **25**(17) (1997) 389–402