Linköping Electronic Articles in
Computer and Information Science
Vol. 6(2001): nr 13

Are grammatical representations useful for learning from biological sequence data? - a case study

C.H. Bryant, S.H. Muggleton,

A. Srinivasan, A. Whittaker,

S. Topp, and C. Rawlings

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/2001/013/

Published on August 30, 2001 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. 6(2001): nr 13. http://www.ep.liu.se/ea/cis/2001/013/. *August 30, 2001.*

This URL will also contain a link to the author's home page.

Abstract

This paper investigates whether Chomsky-like grammar representations are useful for learning cost-effective, comprehensible predictors of members of biological sequence families. The Inductive Logic Programming (ILP) Bayesian approach to learning from positive examples is used to generate a grammar for recognising a class of proteins known as human neuropeptide precursors (NPPs). Collectively, five of the co-authors of this paper, have extensive expertise on NPPs and general bioinformatics methods. Their motivation for generating a NPP grammar was that none of the existing bioinformatics methods could provide sufficient cost-savings during the search for new NPPs. Prior to this project experienced specialists at SmithKline Beecham had tried for many months to hand-code such a grammar but without success. Our best predictor makes the search for novel NPPs **more than 100 times more efficient** than randomly selecting proteins for synthesis and testing them for biological activity. As far as these authors are aware, this is both the first biological grammar learnt using ILP and the first real-world scientific application of the ILP Bayesian approach to learning from positive examples.

A group of features is derived from this grammar. Other groups of features of NPPs are derived using other learning strategies. Amalgams of these groups are formed. A recognition model is generated for each amalgam using C4.5 and C4.5rules and its performance is measured using both predictive accuracy and a new cost function, *Relative Advantage* (RA). The highest RA was achieved by a model which includes grammar-derived features. This RA is significantly higher than the best RA achieved without the use of the grammar-derived features. Predictive accuracy is not a good measure of performance for this domain because it does not discriminate well between NPP recognition models: despite covering varying numbers of (the rare) positives, all the models are awarded a similar (high) score by predictive accuracy because they all exclude most of the abundant negatives.

Affiliations

S.H. Muggleton[1] C.H. Bryant[2]
Department of Computer Science, University of York,
Heslington, York, YO10 5DD, UK.

A. Srinivasan
Oxford University Computing Laboratory, Wolfson Building,
Parks Road, Oxford OX1 3QD, UK.

A. Whittaker[3] Simon Topp Chris Rawlings[4]
SmithKline Beecham, New Frontiers, Science Park, Third
Avenue, Harlow, Essex CM19 5AW, UK.

[1]Current address: Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London SW7 2BZ.

[2]Current address: School of Computer and Mathematical Sciences, The Robert Gordon University, St Andrew Street, Aberdeen, AB25 1HG, Scotland, UK.

[3]Current address: Psygnosis Ltd., 37, Kentish Town Road, London NW1 8NX, UK.

[4]Current address: Oxagen Ltd., 91, Milton Park, Abingdon. OX14 4RY, UK.

1 Introduction

This paper attempts to answer, by way of a case-study, the question of whether grammatical representations are useful for learning from biological sequence data. We address the question with experimental results that significantly contradict the following null hypothesis.

Null hypothesis: The most cost-effective, comprehensible multi-strategy predictors of human neuropeptide precursors do not employ a context-free definite-clause-grammar.

Multi-strategy learning [19] aims at integrating multiple strategies in a single learning system, where strategies may be inferential (e.g. induction, deduction etc) or computational. Computational strategy is defined by the representational system and the computational method used in the learning system (e.g. decision tree learning, neural network learning etc).

A grammar for a language tells us whether a sentence is properly formed. Noam Chomsky, a founder of formal language theory, provided an initial classification of language types. Those readers requiring an introduction to formal grammars or this hierarchy are referred to [17].

We obtain results which significantly contradict the null hypothesis as follows. A grammar is generated for a particular class of biological sequences. A group of features is derived from this grammar. Other groups of features are derived using other learning strategies. Amalgams of these groups are formed. A recognition model is generated for each amalgam using C4.5 and C4.5rules. The results significantly contradict the null hypothesis because:-

1. the best performance achieved using any of the models which include grammar-derived features is higher than the best performance achieved using any of the models which do not include the grammar-derived features;
2. this increase is shown to be statistically significant;
3. the best model which includes grammar-derived features is sufficiently more comprehensible than the best 'non-grammar' model.

Performance is measured using a new cost function, Relative Advantage (RA). Appendix A defines RA and explains why it is used in preference to other performance measures. A method of estimating the RA of a recognition model is presented which subsequently allows the statistical significance of the difference between the RA of two models to be gauged.

The domain of the case study is the recognition of a class of proteins known as human neuropeptide precursors (NPPs). These proteins have considerable therapeutic potential and are of widespread interest in the pharmaceutical industry (see Section 3). Our best multi-strategy predictor of NPPs employs a context-free definite-clause-grammar.

An Inductive Logic Programming (ILP) [23] system is used to generate a grammar for NPPs. As far as these authors are aware, this is the first attempt to generate a grammar for a biological domain using ILP. ILP is the area of Artificial Intelligence which deals with the induction of hypothesised predicate definitions from examples and background knowledge. Logic programs are used as a single representation for examples, background knowledge and hypotheses. For a recent overview of ILP issues and results see [21].

Most ILP systems require both a set of positive examples of the concept to be learnt and a set of negative examples. However it is not possible to identify a large, unbiased set of negative examples of NPPs with certainty because there will be proteins which have yet to be recognised scientifically as a NPP. Therefore advantage was taken of the ILP Bayesian approach to learning from positive examples [24]. This approach does not require a set of negative examples. It is able to learn a concept from a set of positive examples and a set of examples sampled at random. As far as these authors are aware, this is the first real-world scientific application of this approach.

The paper is organised as follows. Section 2 describes the role of sequence information in molecular biology and previous techniques for learning from it, including grammatical inference. This section reviews grammatical inference from the viewpoint of learning cost-effective, comprehensible predictors of members of biological sequence families. Those readers interested in previous theoretical results and applications to natural language are referred to [34] and the references therein. Section 3 introduces neuropeptide precursor recognition, the domain of the case-study. Sections 4 and 5 detail the experimental materials, methods and results. Section 6 is the Discussion. Appendix A describes the new cost function Relative Advantage (RA). Appendix B includes the production rules generated by CProgol. Appendix C includes our best multi-strategy predictor of NPPs.

2 Sequence Data in Biology

Research in the biological and medical sciences is being transformed by the volume of data coming from projects which will reveal the entire genetic code (genome sequence) of *Homo sapiens* as well as other organisms. Once complete, these projects should help us understand the genetic basis of human disease. The growth in the volume of data and improvements in software for interpreting this information has increased interest in the use of computational methods for identifying genes involved in human disease [31]. Knowing the genes implicated in a disease identifies the proteins that they code for and possibly suggests the biochemical processes that may be influencing the development of the disease. This information is crucial for the generation of the experimental reagents needed for the development of new drugs and explains the widespread investment by the biotechnology and pharmaceutical industries in bioinformatics staff and technologies [18, 39].

Recent announcements indicate a commitment to complete sequencing of the entire human genome during the year 2000, through accelerated international funding of public research [28]. This initiative has promoted the development of technology to generate raw (uninterpreted) gene sequence data at a rate of at least 1 Gigabase of new DNA per year. Once the full human genome sequence is available, it will not be long before the nucleotide sequence of every human gene is known. The amino acid sequences of the proteins encoded by these genes can then be deduced. Current estimates of the number of genes in the human genome vary greatly, but tend to average around 60,000. If different, but similar biological sequences are believed to have arisen by evolution from a common ancestor, the proteins or genes are said to be homologous to each other. For a significant portion of these deduced protein sequences, a function can be inferred due to the homology of the sequence to another known protein. However, given the large numbers of new protein sequences expected to be solved in the near future there will be a great many for which no clear homologues of known function exist.

Data deposition at this rate will challenge all aspects of the existing genomic information processing infrastructure in both commercial and academic research sectors. Current state of the art computational sequence interpretation methods are not capable of solving the sequence to function problem for all new proteins, and the development of new techniques is still required.

A significant challenge in the analysis and interpretation of genetic sequence data is therefore the accurate recognition of patterns within the data that are diagnostic for known structural or functional features within the protein. The language of genes is written in a simple alphabet {`A, C, G, T`} representing the four DNA base codes. The language of proteins uses a twenty character alphabet {`A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y`} representing amino acid residues. These residues are encoded in genes by successive DNA base triplets. At their simplest, these patterns can be described as regular expressions. Many features can be described through the use of regular expressions and a database PROSITE [3] is available in which these patterns are curated. A PROSITE pattern such as:
`[AC]-x-V-x(4)-{ED}`
is translated as:
`[A or C]-any-V-any-any-any-any-{any but E or D}`
A more extensive example of a PROSITE pattern is that for the family of proteins called short-chain dehydrogenases (enzymes involved in cell metabolism). The PROSITE pattern that includes two perfectly conserved residues, a tyrosine (Y) and a lysine (K) is:
`[LIVSPADNK]-x(12)-Y-[PSTAGNCV]-[STAGNQCIVM]-[STAGC]-K-{PC}-[SAG-FYR]-[LIVMSTAGD]-x(2)-[LIVMFYW]-x(3)-[LIVMFYWGAPTHQ]-[GSACQRHM]`

There are, however, limitations inherent in the use of simple regular expressions as a representation of biological sequence patterns. In recent years attention has shifted towards both the use of neural network approaches (see [5]) and to probabilistic models, in particular hidden Markov models (see [11]). Both these methods directly address the extent of variation in the biological world. A significant advantage of both the probabilistic methods and the neural network approaches is that they are complemented with well-established methods for training models from examples. Training regimes generally (but not always) require that the sequences in the training set be arranged so that those regions of the sequence that have been conserved through evolution are aligned in the same column. The accurate multiple alignment of biological sequences has been the subject of much research and discussion [10]), and is considered by many to be a solved problem. However, it relies on the assumption that the sequences to be aligned show some homology at the sequence level with each other. Complex biological signals also require complex models and it is often the case that considerable expertise is required in the selection of the optimal neural network architecture or hidden Markov model before training can take place.

A general linguistic approach to representing the structure and function of genes and proteins has intrinsic appeal as an alternative approach to probabilistic methods because of the declarative and hierarchical nature of grammars. Searls [35] has undertaken the most thorough analysis of the linguistic classification of genetic grammars starting with the Definite Clause Grammar (DCG). Searls proposes a String Variable Grammar (SVG) extension to a DCG to provide features necessary for representing higher-order interactions among genetic sequence elements found in nucleic acids such as non-linear features found in RNA pseudoknots and other secondary structures formed as a result of internal nucleic acid base-pairing.

While linguistic methods have provided some interesting results in the recognition of complex biological signals [36] general methods for learning new grammars from example sentences are much less developed. Brazma has reviewed the development of methods for the automatic discovery of biological patterns [6] much of which has taken place in the context of building databases of sequence motifs such as PROSITE [3], BLOCKS [13] and PFAM [38, 37]. Abe & Mamitsuka proposed a method for predicting the protein secondary structure of a given amino acid sequence using a training algorithm for a class of stochastic tree grammars [1].

We considered it valuable to investigate the application of Inductive Logic Programming methods to the discovery of a language that would describe a particularly interesting class of sequences – neuropeptide precursor proteins (NPP). They are highly variable in length and undergo specific enzymatic degradation (proteolysis) before the biologically active short peptides (neuropeptides) are released. Unlike enzymes or structural proteins, NPPs tend to show almost no overall sequence similarity with the exception of some evidence for common ancestry within certain groups. Roughly, the 50 human neuropeptide precursors currently known contain about 140 known cleaved peptides. These peptides belong to at least 40 different neuropeptide families, with only 1-5 members per family. Prosite motifs do exist for some of the more highly populated families, but these are based on the cleaved peptide and not the precursor. These motifs could be used to discover new members of a neuropeptide family, but they will never detect novel families of neuropeptides. It is believed that there are a great many more novel subgroups yet to be discovered. This confounds pattern discovery methods that rely on multiple sequence alignment and recognition of biological conservation. As a consequence NPPs pose a particular challenge in sequence pattern discovery and recognition.

3 Neuropeptide Precursor Proteins

Neuropeptides are an important group of short proteins that act as neurotransmitters mediating the passage of signals within the central nervous system (CNS) and between the CNS and the rest of the body. The term neuropeptide was first introduced in 1971 by D. de Weid [15] to describe fragments of hormones that produced behavioural changes when injected, but lacked the activity of the intact hormone. More recently the term has been accepted to cover peptides united by a number of common features, including their tissue expression (brain, nervous tissue, secretory cells from organs such as gut, heart, lungs, placenta etc) metabolism, secretion, biosynthesis and high potency [15].

Drug molecules work by interacting with target sites within the body. These

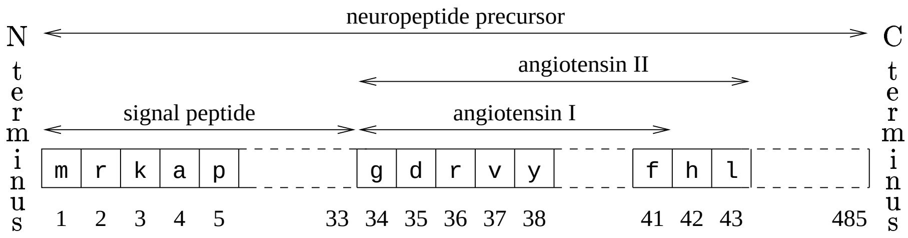

Figure 1: A neuropeptide precursor sequence containing the Angiotensin neuropeptides. A precursor will always contain exactly one signal peptide. The number of neuropeptides can vary.

sites commonly are protein molecules, either enzymes or receptors. By interaction with these protein molecules, drugs can modulate their actions and generally suppress undesirable biochemical reactions. Neuropeptides exert their biological actions through binding as ligands to specific receptors. The term ligand is used for molecules which bind to the target site. (A ligand might be highly active against the target, but not a 'drug', because of a lack of other required properties such as metabolic stability or safety.)

Active research has increased the number of mammalian neuropeptides from about 18 in 1978 to more than 80 by 1999. However, despite all these efforts, the biology of many of these neuropeptides as well as their interactions with their receptors remain to be elucidated. The receptors of some neuropeptides have not yet been identified and there are some orphan receptors with as yet unidentified ligands. The in-vitro pairing of a novel receptor with its ligand is a critical first step in understanding the mechanism of disease. Thus novel neuropeptides and their orphan receptors have considerable therapeutic potential and are of widespread interest in the pharmaceutical industry.

Neuropeptides are subsequences of *neuropeptide precursor* sequences. An example of a neuropeptide precursor is shown in Figure 1. This neuropeptide precursor contains the neuropeptides Angiotensin I and Angiotensin II. A diagrammatic representation of several other precursors is shown in Figure 2. Precursors may contain either a single neuropeptide, multiple copies of the same neuropeptide or several different neuropeptides. These can occur consecutively in the precursor or can be separated by large stretches of *filler* peptide which is believed to play a purely structural role. Neuropeptide precursors contain a short prefix of residues called a *signal peptide* of about 20–30 amino acids (aa) in length. The known precursors range in length from 70 to 600 aa, and the cleaved peptides range from 3-200 aa. It is this huge variation in length, sequence and internal organisation that makes neuropeptide precursors difficult to use when searching for novel remote homologues using sequence database searching methods (e.g. BLAST). They also confound typical multiple sequence alignment methods used to identify conserved features among functionally related sequences.

Many proteins are cleaved and trimmed after synthesis. For example, digestive enzymes are synthesised as inactive precursors that can be stored safely in the pancreas. After being released into the intestine, these precursors become activated by cleavage. Neuropeptide precursors undergo this 'splitting' process. The signal peptide targets the protein for secretion through a cell membrane where it is then cleaved from the precursor. The

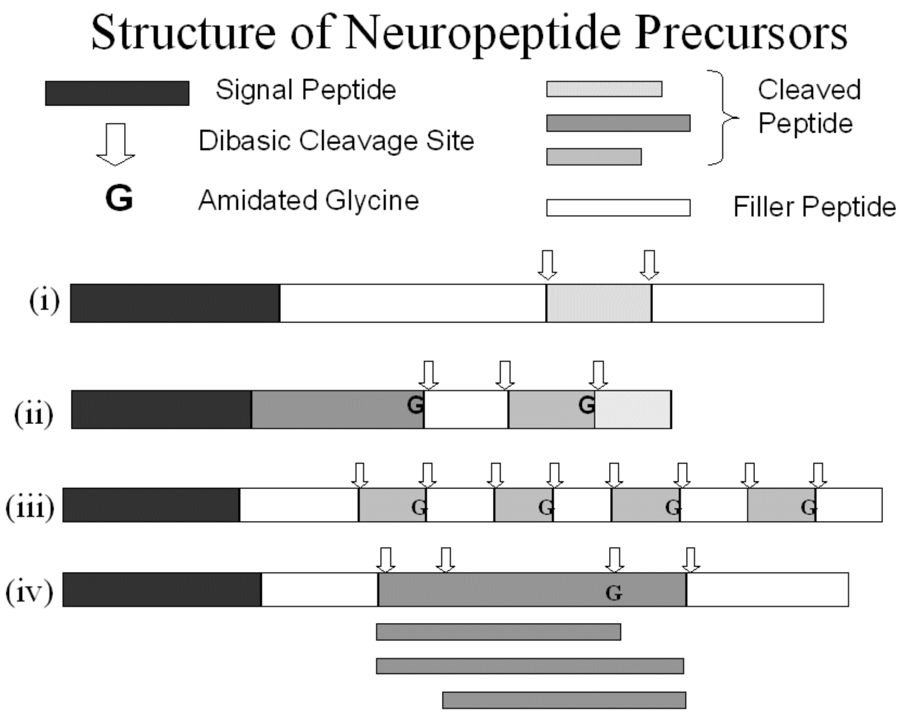

Figure 2: Some of the different configurations of neuropeptides known to occur within human precursors. Precursor (i) shows a single cleaved peptide flanked by filler of unknown function. Precursor (ii) shows three different cleaved peptides, two of which are adjacent to each other, whilst the other is separated by filler. Precursor (iii) contains the same short peptide repeated 4 times. Precursor (iv) shows that cleavage can occur selectively, giving alternative termini to the released peptides.

remainder of the precursor is further cleaved to release the neuropeptide. Within the precursor, the location of the cleavages of the signal sequence and the neuropeptides are referred to as *cleavage sites*.

To our knowledge there has been no previous attempt to computationally predict or recognise neuropeptides. Several investigators have statistically analysed various features of neuropeptides and their precursors [8, 32, 33, 4]. Other investigators have developed methods for the identification of protein sorting signals and the prediction of their cleavage sites [7, 26, 27]. However the recognition of signal peptides only solves part of the problem of how to recognise neuropeptide precursors. Within SmithKline Beecham our previous work in predicting novel neuropeptide precursors has centred on the use of regular expression searching of translated Expressed Sequence Tag (EST) databases of gene fragments. The results of this approach were limited by the rigidity of the regular expressions, the high frequency of errors in EST sequences [14], and their relatively short length. The ultimate goal of this project will be to differentiate novel neuropeptides from the wealth of other new sequences produced by the completion of the human genome sequencing project. This is a task that current sequence analysis tools have so far failed to solve.

4 Experiment One

This section describes an experiment whose results significantly contradict the null hypothesis (see Section 1). The section begins by describing the materials (data, background knowledge and machine learning systems) used in the experiment. This is followed by an account of the three steps of the experimental method. Finally the section ends with the presentation and analysis of the results.

4.1 Materials

4.1.1 Data

The data was taken from the SWISS-PROT database [2]. SWISS-PROT is an annotated protein sequence database established in 1986 and maintained, with collaborators, by the Department of Medical Biochemistry of the University of Geneva. It can be accessed at `http://www.expasy.ch/sprot/sprot-top.html`.

Our data-set comprises a subset of positives i.e. known NPPs and a subset of randomly-selected sequences. It is not possible to generate a large, unbiased set of negative examples because there will be proteins which have yet to be recognised scientifically as a NPP. The characteristics of the two subsets of sequences are as follows.

Positives This subset contains all of the 44 known NPP sequences that were in SWISS-PROT in Spring 1997, the time the data-set was prepared (see Table 8). The SWISS-PROT identifiers of these 44 sequences are listed in Tables 1 and 2.

All but three of these are human proteins. The three non-human sequences were included as the human equivalent had not been discovered and they were considered to be important examples. They

Table 1: SWISS-PROT identifiers for the NPPs in the training-set.

P01019	P01042	P01156	P01178	P01185	P01189	P01210	P01213	P01258
P01270	P01275	P01279	P01282	P01286	P01298	P01303	P05060	P05305
P06881	P07491	P08858	P08949	P10082	P10645	P12272	P14138	P16860
P18509	P20366	P20382	P20800	P21591	P22466	P35318		

Table 2: SWISS-PROT identifiers for the NPPs in the test-set.

P01148	P01160	P01166	P06307	P06850	P10997	P13521	P23582	P20396
P01350								

are expected to be very closely related to the human and are possibly a reasonable model for humans.

10 of the 44 precursors were selected to constitute part of the test-set. These sequences are unrelated by sequence homology to the remaining 34.

Randoms This subset contains all of the 3910 full length human sequences in SWISS-PROT in Spring 1997.

1000 of the 3910 randoms were reserved for the test-set.

The data-set is available at
`ftp://ftp.cs.york.ac.uk/pub/aig/Datasets/neuropeps/`

4.1.2 Machine Learning Systems

The propositional learning was performed using the decision-tree learner C4.5 (Release 8) in conjunction with the companion program C4.5rules that constructs rules from a tree built by C4.5 [30]. The grammar learning was performed using CProgol [20] version 4.4 which is available from
`ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/progol4.4`.

4.1.3 Background Knowledge

During both the generation of the grammar using CProgol and the generation of propositional rule-sets using C4.5 and C4.5rules we adopt the background information used in [22] to describe physical and chemical properties of the amino acids (see Table 3).

4.2 Method

The method may be summarised as follows:-

1. A grammar is generated for NPP sequences using CProgol (see Section 4.2.1).
2. A group of features is derived from this grammar. Other groups of features are derived using other learning strategies. (See Section 4.2.2).

Table 3: Physical and chemical properties of the amino acids.

Physicochemical Property	Amino acids with property
Hydrophobic	H,W,Y,F,M,L,I,V,C,A,G,T,K
Very hydrophobic	A,F,G,I,L,M,V
Hydrophilic	S,E,Q,R,D,N
Electropositive	R,K,H
Electronegative	D,E
Neutral	A,C,F,G,I,L,M,N,P,Q,S,T,V,W,Y
Large	Q,E,R,K,H,W,Y,F,M,L,I
Small	P,V,C,A,G,T,S,N,D
Tiny	A,G,S
Polar	Y,T,S,N,D,E,Q,R,K,H,W
Aliphatic	L,I,V
Aromatic	H,W,Y,F
Hydrogen donor	W,Y,H,T,K,C,S,N,Q,R
Hydrogen acceptor	Y,T,C,S,D,E,N,Q

3. Amalgams of these groups are formed. A rule-set is generated for each amalgam using C4.5 and C4.5rules and its performance is measured using $Mean\ RA$. This is a new cost function which is described in Appendix A. The null-hypothesis (see Section 1) is then tested by comparing the $Mean\ RA$ achieved from the various amalgams. (See Section 4.2.3).
4. A hidden Markov model (HMM) is generated for NPP sequences and its $Mean\ RA$ is measured (see Section 4.2.4).

4.2.1 Grammar Generation

A *NPP grammar* contains rules that describe legal neuropeptide precursors. Figure 3 shows an incomplete example of such a grammar, written as a Prolog program. This section describes how production rules for signal peptides and neuropeptide starts, middle-sections and ends were generated using CProgol. These were used to complete the context-free definite-clause-grammar structure shown in Figure 3. The start and end represent cleavage sites and the middle-section represents the mature neuropeptide i.e. what remains after cleavage has taken place.

The production rules to be learnt by CProgol contains dyadic predicates of the form `p(X,Y)`, which denote that property `p` began the sequence `X` and is followed by a sequence `Y`. To learn such rules from the the training-set, CProgol was provided with the following extensional definitions:

Precursor data. Using details of the start and finishing positions for signal peptides and neuropeptides it was possible to generate examples of non-terminals as below:

`signalpep(S,[])` where `S` is a list of precursor residues constituting the signal peptide.

`start(S,[])` where `S` is a list of residues constituting the start of a neuropeptide. The length of `S` was taken to be 2 residues.

```
npp(A,B):- signal(A,C),
           star(C,D),
           neuro_peptide(D,E),
           star(E,B).
signal(A,C):- ...
neuro_peptide(D,E):- start(D,F),
                     middle(F,G),
                     end(G,E).
start(D,F):- ...
middle(F,G):- ...
end(G,E):- ...
```

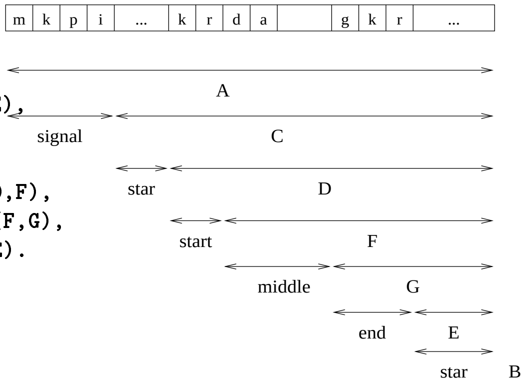

Figure 3: Grammar rules describing legal NPP sequences. The rules comply with Prolog syntax. $npp(X, Y)$ is true if there is a precursor at the beginning of the sequence X, and it is followed by a sequence Y. The other dyadic predicates are defined similarly. $star(X, Y)$ is true if, at the beginning of the sequence X, there is some sequence of residues whose length is not specified and which is followed by another sequence Y. Definitions of the predicates denoted by '...' are to be learnt from data of known NPP sequences.

`middle(S,[])` where `S` is a list of residues constituting the middle of a neuropeptide. The starting residue for `S` is the first residue after the end of the sequence for `start/2` above. The end of `S` was taken to be 3 residues from the last position of the neuropeptide.

`end(S,[])` where `S` is a list of 3 precursor residues constituting the end of a neuropeptide. `S` commences with the residue after the end of the sequence for `middle/2` above.

Random data. One random example for each of `signalpep/2`, `start/2`, `middle/2` and `end/2` was generated from each sequence in the set of randoms sequences. Random examples are distinguished by the prefix `*`.

`*signalpep(S,[])` where `S` is a list of residues starting at the first position in the sequence. The length of `S` is obtained from drawing randomly from the distribution of signal peptide lengths of NPPs in the training data.

`*start(S,[])` where `S` is a pair of sequence residues. The starting residue is randomly chosen, and is ensured not to conflict with the sequence chosen for `*signalpep/2` above.

`*middle(S,[])` where `S` is a list of residues starting after the end of the sequence for `*start/2` above. The length of `S` is obtained by drawing randomly from the length of neuropeptide middle-sections of precursors in the training data.

`*end(S,[])` where `S` is a list of 3 residues starting at the end of the sequence terminating the definition of `*middle/2` above.

CProgol was provided with definitions of the non-terminals `star/2` and `run/3` (see Table 14). `star/2` represents some sequence of unnamed residues whose length is not specified. `run/3` represents a run of residues which share a specified property.

The grammar-based approach presents a powerful method for describing NPPs, as it allows for the natural inclusion of existing biochemical knowledge. Prior knowledge of NPPs suggested that the following subsequences may be important: `KR`; `GKR` and `GRR`. The subsequences `KR` and `GKR` are established proteolytic cleavage sites found in NPPs; `GRR` is a relatively common alternative cleavage site to `GKR`. Pilot experiments suggested that the following patterns may be significant:

`K,positive`;
`positive,positive`;
`Y,very_hydrophobic`;
`hydrophilic,a_gap_of_some_residues,M,negative`;
`HP`;
`WMDF`.

All these subsequences and patterns were coded as Prolog predicates and included as background knowledge (see Table 14).

Other pilot experiments on the training data showed that the accuracy of CProgol's grammar was higher with certain restrictions on the length of NPPs, signal peptides and neuropeptides. Specific constraints were obtained by progressively checking the following: all lengths less than the mean length on training data; lengths that are within $1, 2, \ldots$ standard deviation of the mean on training data. This resulted in the following additional restrictions: (1) NPP lengths not to exceed 200 residues; (2) signal peptide lengths to be between 19 and 29 residues; and (3) middle-sections of neuropeptides lengths to be between 4 and 52 residues. These constraints only affect the values of features derived from the grammar. They do not constrain the value of the sequence length feature described at the end of Section 4.2.2.

Appendix B shows the mode and type declarations, settings and prune predicates that were needed to enable CProgol to complete the grammar shown in Figure 3. Table 13 shows the production rules that were generated.

4.2.2 Feature Groups

The grammar features Predictions about a NPP sequence can be made by parsing it using the NPP grammar. The values of the features shown in Table 4 were obtained by such parses. Note that whenever the grammar predicts that a sequence is *not* a NPP, all of the features are assigned the value zero .

The SIGNALP features Each feature in this group is a summary of the result of using the SIGNALP program on a sequence. The SIGNALP program [27] represents the pre-eminent automated method for predicting the presence and location of N-terminal signal peptides. SIGNALP is available on the web at `http://www.cbs.dtu.dk/services/SignalP`. The technique used combines the predictions of two different neural networks groups – one that recognises cleavage sites, and the other that identifies signal peptides. When provided with a sequence of N-terminal residues, the following are reported as summaries: (a) C scores: which consist of the maximum value of the score from the cleavage-recogniser, the position in the sequence where this value is achieved, and a nominal 'y' or 'n' denoting the answer to whether a

Table 4: The grammar group of features.

Feature	Description
`gram_pred`	A boolean which indicates whether the grammar predicts a sequence to be a NPP or not.
`gram_sig_l`	Length of the signal peptide.
`gram_np_l`	Length of the neuropeptide.
`gram_first`	Position of the first residue in the neuropeptide.
`gram_last`	Position of the last residue in the neuropeptide.
`gram_np_start_first`	The first residue in the neuropeptide or one of its properties.
`gram_np_start_second`	The second residue in neuropeptide or one of its properties.
`gram_np_end_first`	The first residue/property/star in the body of the end rule.
`gram_np_end_second`	The second residue/property/star in the body of the end rule.
`gram_np_end_third`	The third residue/property/star in the body of the end rule.

cleavage site is present; (b) S scores: the corresponding values from the signal-peptide recogniser; (c) Y scores: a score that combines the C and S scores; and (d) Mean scores: a mean of the S-score and S-conclusions from the N-terminal end to the predicted cleavage site. For the experiments here, SIGNALP was provided with 50 amino acids from the N-terminal for each sequence. The summaries were extracted and represented by 11 features shown in Table 5.

The proportions features Each feature in this group is a proportion of the number of residues in a given sequence which either are a specific amino-acid or which have a specific physicochemical property of an amino-acid. Hence there is one such feature for each of the a) 20 amino acids b) properties shown in Table 3.

The sequence length feature This feature is the length of the sequence. In the remainder of this paper this feature will be referred to as `length`.

4.2.3 Propositional Learning

The training and test data sets for C4.5 were prepared as follows.

1. Recall from Section 4.1.1 that our data comprises 44 positives and 3910 randoms. 40 of the 44 positives occur in the set of 3910 randoms. As C4.5 is designed to learn from a set of positives and a set of negatives, these 40 positives were removed from the set of randoms. Of the 40 positives which are in the set of randoms, 10 are in the test-set. Hence the set of $(3910 - 40)$ sequences were split into a training-set of $(2910 - 30 = 2880)$ and a test-set of $(1000 - 10 = 990)$.
2. Values of the features were generated for each training and test sequence. Each sequence was represented by a data vector comprised

Table 5: The SIGNALP group of features

Feature	Description
`sigp_cmax`	Maximum SIGNALP C score
`sigp_cmaxpos`	Position where maximum SIGNALP C score is achieved
`sigp_cconcl`	SIGNALP C score conclusion ('y' or 'n')
`sigp_ymax`	Maximum Y score reported by SIGNALP
`sigp_ymaxpos`	Position where maximum SIGNALP Y score is achieved
`sigp_yconcl`	SIGNALP Y score conclusion ('y' or 'n')
`sigp_smax`	Maximum SIGNALP S score
`sigp_smaxpos`	Position where maximum SIGNALP S score is achieved
`sigp_sconcl`	SIGNALP S score conclusion ('y' or 'n')
`sigp_smean`	SIGNALP mean of S scores to cleavage site
`sigp_smeanconcl`	SIGNALP mean S score conclusion

of these feature values and 1 class value ('1' to denote a NPP and '0' otherwise).

3. Finally to ensure that there were as many '1' sequences as '0' sequences a *training*-set of 2880 NPPs was obtained by sampling with replacement. Thus the training data-set input to C4.5 comprised (2×2880) examples. (No re-adjusting was done on the test data.)

Amalgams of the feature groups described in the previous section were formed. The amalgams are listed in Table 6. The following procedure was followed for each one:-

1. training and test sets were prepared as described above;
2. a decision tree was generated from the training-set using C4.5;
3. a rule-set was generated from this tree using C4.5rules;
4. a 2×2 contingency table was drawn-up based on the predictions of this rule-set on the test-set;
5. $Mean\ RA$ was estimated as described in Appendix A.3.

The default settings of C4.5 and C4.5rules were used.

The contradiction of the null hypothesis was then attempted by testing whether:-

- the $Mean\ RA$ of the best model which includes grammar-derived features was higher than the best performance achieved using any of the models which do not include the grammar-derived features.
- such an increase was statistically significant. Estimates of $Mean\ RA$ were compared using the statistical method described in Appendix A.4.
- the best model which includes grammar-derived features was sufficiently more comprehensible than the best 'non-grammar' model.

4.2.4 Hidden Markov Model Comparison

A HMM for NPPs was generated and tested using HMMER[5] version 2.1.1 [12] which is available from `http://hmmer.wustl.edu/`.

CLUSTAL W (1.8) [40] was used to align the positive sequences in the training-set. The `hmmbuild` program of HMMER was then used to generate a HMM from the CLUSTAL W alignment. The resulting HMM and the `hmmsearch` program of HMMER were then used to search for NPPs in the test-set. The default settings of CLUSTAL W and HMMER were used. The $Mean\ RA$ of the HMM was estimated based on the predictions on the test-set.

4.3 Results and Analysis

Table 13 shows the grammar that was generated using CProgol. The grammar is very rich in terms of non-terminals. All but one of the non-terminals which represent the properties of residues listed in Table 3 appear in the grammar. The grammar also includes `star/2`, `run/3` and three of the six non-terminals which represent the patterns mentioned in Section 4.2.1. The non-terminals `yvh` and `pp`, which represent the patterns `Y,very_hydrophobic` and `positive,positive`, both appear twice. The third non-terminal which appears is `hmn`, which corresponds to the pattern `hydrophilic,a_gap_of_some_residues,M,negative`.

Table 6 shows that predictive accuracy is not a good measure of performance for this domain because it does not discriminate well between the amalgams: despite covering varying numbers of (the rare) positives, all the models are awarded a similar (high) score by predictive accuracy because they all exclude most of the abundant negatives.

Table 6 shows the $Mean\ RA$ for both the hidden Markov model and for each amalgam of feature groups. The $Mean\ RA$ of the HMM is zero. The highest $Mean\ RA$ (107.7) was achieved by one of the grammar amalgams, namely the 'Proportions + Length + SIGNALP + Grammar' amalgam. The best $Mean\ RA$ achieved by any of the amalgams which do not include the grammar-derived features was the 49.0 attained by the 'Proportions + Length' amalgam.

The $\sum_{M=57}^{90} RA$ for the 'Proportions + Length + SIGNALP + Grammar' amalgam was 3661.376. The $\sum_{M=57}^{90} RA$ for the 'Proportions + Length' amalgam was 1666.733. For the amalgams 'Proportions + Length + SIGNALP + Grammar' and 'Proportions + Length', $\hat{\mu}_D = 1994.643$ and $\hat{\sigma}_D/\sqrt{n} = 2.081$. This difference is statistically significant: substituting these values of $\hat{\mu}_D$ and $\hat{\sigma}_D/\sqrt{n}$ into Equation 15 shows that $p(d < 0)$ is well below 0.0001.

If one searches for a NPP by randomly selecting sequences from SWISS-PROT for synthesis and subsequent biological testing then, at most, only one in every 2408 sequences tested is expected to be a novel NPP. This follows from the fact that number of sequences in SWISS-PROT = 79449, the most probable number of *novel* NPPs in SWISS-PROT $= 90 - 57 = 33$ (see Table 8) and $33/79449 = 1/2408$. Using our best recognition model as a filter makes the search for a NPP far more efficient. Approximately one in every 22 of the randomly selected SWISS-PROT sequences which pass through

[5] The validity of the result was checked with the author of HMMER.

Table 6: Estimates of $Mean\ RA$ and predictive accuracy of both the HMM and the decision trees generated from the amalgams of the feature groups. $Mean\ RA$ was estimated using the method described in Section A.3.

Predictor	$MeanRA$	Predictive Accuracy (%)
Hidden Markov Model	0	99.0 $\pm$ 0.3
Only props	0	96.7 $\pm$ 0.6
Only Length	1.6	91.8 $\pm$ 0.9
Only SignalP	11.7	98.1 $\pm$ 0.4
Only Grammar	10.8	97.0 $\pm$ 0.5
Props + Length	49.0	98.6 $\pm$ 0.4
Props + SignalP	15.0	98.3 $\pm$ 0.4
Props + Grammar	31.7	98.2 $\pm$ 0.4
SignalP + Grammar	0	98.6 $\pm$ 0.4
Length + Grammar	0	96.2 $\pm$ 0.6
Length + SignalP	34.4	98.7 $\pm$ 0.4
Length + SignalP + Grammar	0	98.0 $\pm$ 0.4
Props + Length + SignalP	29.2	98.7 $\pm$ 0.4
Props + Length + Grammar	33.2	98.5 $\pm$ 0.4
Props + SignalP + Grammar	15.0	98.3 $\pm$ 0.4
Props + Length + SignalP + Grammar	107.7	99.0 $\pm$ 0.3

our filter is expected to be a novel NPP. This can be seen from the following simple calculation. Rearranging Equation 2 gives $Pr(NPP \mid Rec) = RA \times Pr(NPP)$. Substituting in the $MeanRA$ for the best recognition model gives $Pr(NPP \mid Rec) = 107.688 * (90/79449) = 1/8.2$. Multiplying 1/8.2 by the proportion of NPPs in SWISS-PROT which are novel (33/90) gives approximately 1/22.

Appendix C lists the complete rule-sets for the amalgams 'Proportions + Length + SIGNALP + Grammar' and 'Proportions + Length'. The rules that were generated from the 'grammar amalgam' suggest that the NPP grammar is useful for learning from NPP sequence data. Nine of the 25 rules include a *grammar-derived* feature. These rules refer to a variety of the *grammar-derived* features:-

- whether the grammar predicts the existence of an neuropeptide start (e.g. see Rule 14 in Figure 5);
- the first residue in the neuropeptide (e.g. see Rules 20 and 21 in Figures 5 and 6 respectively);
- the position of the first residue in the neuropeptide (e.g. see Rule 6 in Figure 5);
- the property of the third from last residue in the neuropeptide (e.g. see Rule 17 in Figure 5).
- the length of the signal peptide;

Table 7: SWISS-PROT identifiers for the additional 13 NPPs which had been added to SWISS-PROT by May 1999.

P10092	P23435	O00230	P09681	O43555	P07492	P48645	P01138	P20783
P02818	Q13519	Q00072	P55089					

Our method did not try to remove the potential redundancy between values of some of the SIGNALP features and grammar features. The results listed in Table 6 justify this. It is of interest to note that the grammar-derived length of signal peptide is used frequently by C4.5 (see Rules 2, 9, 23 and 24), *despite the availability of similar features derived from SIGNALP* (see Table 5).

Finally, it should be noted that two of the rules refer *only* to grammar-derived features (see Rules 20 and 21).

5 Experiment Two

Aim The aim of the second experiment is to demonstrate that overfitting of the NPP sequences did not inadvertently occur during the generation of grammar in the first experiment (see Section 4.2.1). The data-set used in the first experiment contained all of the 44 known NPP sequences in SWISS-PROT in Spring 1997. The second experiment utilises 13 *additional* NPP sequences which had been added to SWISS-PROT by May 1999 (see Table 8). None of these 13 additional NPP sequences were used for training or testing in the first experiment. Indeed the identifiers of these 13 sequences were not known to C.H.B (who performed the second experiment) until after the results of the first experiment were published [25].

Data Two test-sets are used in this experiment.

1. The test-set used in the first experiment i.e. the one which contains 10 of the 44 original NPPs and the 990 'other' sequences.
2. A new test set comprising the 990 'other' sequences from the test-set used in the first experiment and 10 of the 13 additional NPP sequences which had been added to SWISS-PROT by May 1999. The SWISS-PROT identifiers of the additional 13 sequences are listed in Table 7 and the sequences are available at `ftp://ftp.cs.york.ac.uk/pub/aig/Datasets/neuropeps/`. Three of the 13 additional NPP sequences (P10092, P07492 and Q00072) were not used as they are homologues of NPPs in the original training set of 34 NPPs.

Method The $MeanRA$ of the grammar on both the original test-set and the new test-set was measured. Note that it was the $MeanRA$ of the grammar generated by CProgol that was measured as opposed to one of the decision trees generated by C4.5 and C4.5rules. $MeanRA$ was estimated using the method described in Section A.3.

Result The $MeanRA$ of the grammar was the 5.6 for both the original test-set and the new test-set.

Conclusion Overfitting of the original set of NPP sequences did not occur during the generation of the grammar: the performance of the grammar on the new and original data-sets is the same. This result provides further evidence to support our conclusion that we have developed a NPP recognition method which would provide cost-savings during a search for novel NPPs.

6 Discussion

This paper has shown that the most cost-effective, comprehensible multi-strategy predictor of human neuropeptide precursors does employ a context-free definite-clause-grammar.

The ILP Bayesian approach to learning from positive examples was used to generate a grammar for recognising a class of proteins known as human neuropeptide precursors (NPPs). Collectively, five of the co-authors of this paper, have extensive expertise on NPPs and general bioinformatics methods. Their motivation for generating a NPP grammar was that none of the existing bioinformatics methods could provide sufficient cost-savings during the search for new NPPs. Prior to this project experienced specialists at SmithKline Beecham had tried for many months to hand-code such a grammar but without success. Our best predictor makes the search for novel NPPs **more than 100 times more efficient** than randomly selecting proteins for synthesis and testing them for biological activity (see Figure 4). As far as these authors are aware, this is both the first attempt to learn a biological grammar using ILP and the first real-world scientific application of the ILP Bayesian approach to learning from positive examples.

We first published that our best predictor delivers more than a hundred-fold cost-saving in the proceedings of seventeenth international conference on Machine Learning [25]. Since then we have obtained further evidence to support our conclusion that we have developed a NPP recognition method which would provide cost-savings during a search for novel NPPs. We have shown, using NPP sequences which had not been used previously on this project, that overfitting of the original set of NPP sequences did not occur during the generation of the grammar.

A shortcoming of the NPP grammar generated is that it will not recognise all NPPs because it implies that 1) both start and end cleavage sites are compulsory; 2) a mature neuropeptide cannot be adjacent to the signal peptide or at the C-terminus unless it contains a start or an end cleavage site respectively. These restrictions could be removed by adding extra clauses to the definition of the predicates `npp/2` and `neuro_peptide/2` shown in Figure 3. Experiments with a more flexible grammar should therefore form the subject of future work.

In our opinion, the best 'non-grammar' recognition model does not provide any biological insight. However the best recognition model which includes grammar-derived features is broadly comprehensible and contains some intriguing associations that may warrant further analysis. This model is being evaluated as an extension to existing methods used in SmithKline Beecham for the selection of potential neuropeptides for use in experiments to help elucidate the biological functions of G-protein coupled receptors. It is clear however, that the rules of the model are not an optimal representation of

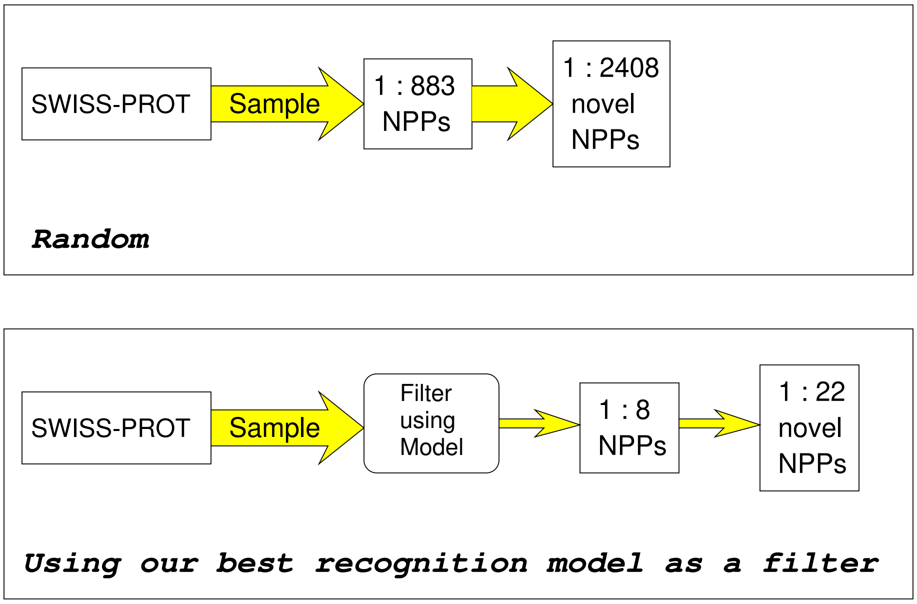

Figure 4: The advantage of using our best recognition model to search for a novel NPP in SWISS-PROT. If one searches for a NPP by randomly selecting sequences from SWISS-PROT for synthesis and subsequent biological testing then, at most, only one in every 2408 sequences tested is expected to be a novel NPP. Using our best recognition model as a filter makes the search for a NPP far more efficient. Approximately one in every 22 of the randomly selected SWISS-PROT sequences which pass through our filter is expected to be a novel NPP. To put this in terms of the economies of the search for new and valuable neuropeptides our best predictor delivers more than a hundred-fold saving.

sequence data and residue properties. A more intuitive (e.g. graphically oriented, sequence centred) display of the meaning of these rules would be required to build tools that the experts in the field would find acceptable.

The new cost function presented in this paper, Relative Advantage (RA), may be used to measure performance of a recognition model for any domain where

1. the proportion of positives in the set of examples is very small.
2. there is no guarantee that all positives can be identified as such. In such domains, the proportion of positive examples in the population is not known and a large, unbiased set of negatives cannot be identified with complete confidence.
3. there is no benchmark recognition method.

In Appendix A we have developed a general method for assessing the significance of the difference between RA values obtained in comparative trials. RA is estimated by summing the estimate of performance on each test-set instance. The method uses a) identically distributed random variables representing the outcome for each instance; b) a sample mean which approaches the population mean in the limit and c) a relatively small sample variance.

7 Contribution of Authors

The co-authors from SmithKline Beecham identified the problem addressed by the case-study, prepared the data-set, provided expertise on NPPs and general Bioinformatics methods and also helped to write the domain-specific aspects of the paper. S.H.M. developed CProgol and the method of generating a NPP grammar. A.S. developed the method of using C4.5 to learn propositional rules from features derived from SIGNALP, proportions and sequence length. C.H.B. devised the set of grammar features, performed the experiments described in this paper and prepared all the sections of this paper except those on the domain. C.H.B. and S.H.M. developed the method for assessing the significance of the difference between the RA of two models and an associated method estimating RA.

8 Acknowledgements

We would like to thank I.S.Gloger, M.Lawrence and R.B.Russell for sharing their knowledge of NPPs and sequence homology. We would like to thank J.Cussens for his help with Section A, M.Turcotte for his advice on HMM software and D.Michie for his comments on this work.

This research was conducted as part of a projected entitled 'Using Machine Learning to Discover Diagnostic Sequence Motifs' supported by a grant from SmithKline Beecham. During part of his time spent writing this article C.H.B. was supported by the EPSRC grant (GR/M56067) entitled 'Closed Loop Machine Learning'. A.S. holds a Nuffield Trust Research Fellowship at Green College, Oxford.

Table 8: Properties of sequences in SWISS-PROT in Spring 1997 and in May 1999.

	Spring 1997	May 1999
Number of sequences	64,000	79,449
Number of known human NPPs	44	57
Most probable number of human NPPs	Not known	90

A Relative Advantage

NPPs are identified either through purely biological means or by screening genomic or protein sequence databases for likely NPPs, followed by biological evaluation. If we wish to go beyond using sequence homology to find new members of the (generally small) NPP families, we need a recognition model for NPPs in general. However if this recognition model is poor then it may not be much better than random sampling of sequence databases (e.g. SWISS-PROT) and the cost-benefit of any experimental evaluation of NPPs found by such a procedure would be prohibitively small.

In developing a general recognition model for human NPPs, we are faced with three significant obstacles.

1. The number of known NPPs in the public domain databases of protein sequence (e.g. SWISS-PROT) is very small in proportion to the total number of sequences. When we developed our method of estimating RA (May 1999), SWISS-PROT contained 79,449 sequences, of which some 57 could definitely be identified as human NPPs.
2. There is no guarantee that all the human NPPs in SWISS-PROT have been properly identified. We estimate there may, in fact be up to 90 NPPs in SWISS-PROT.
3. There is no benchmark method for NPP recognition that can be used to compare any new methods. We must therefore compare our recognition model with random sampling to evaluate success.

This domain requires a performance measure which addresses all of these issues.

Table 8 summarises how some of the properties of SWISS-PROT changed over the duration of the experiments described in this paper. All the RA measurements in this paper are based on the properties as they stood at May 99. When measuring performance using RA there is no requirement that the size of the test data-set is equal to the number of known human NPPs in SWISS-PROT.

A.1 Limitations of Existing Performance Measures

For domains in which positives are rare, predictive accuracy, as it is normally measured in Machine Learning (assuming equal misclassification costs):

- gives a poor estimate of the performance of a recognition model. For instance, if a learner induces a very specific model for such a domain, the predictive accuracy of the model may be very high despite the number of true positives being very small or even zero.

- does not discriminate well between models which exclude most of the (abundant) negatives but cover varying numbers of (the rare) positives. (This was illustrated earlier in this paper - see Table 6.)

For domains in which there is no benchmark recognition method that can be used to compare any new methods, Lift [16] is not the appropriate measure of performance because it does not quantify the reduction in cost in using the predictor versus random sampling. Furthermore, in their paper, Ling & Li gave no explanation of how to assess the significance of the difference between the Lift of two models. ROC curves (or Lorentz diagrams) [29] also do not quantify the reduction in cost in using the predictor versus random sampling.

Therefore we define a *relative advantage* (RA) function which predicts the reduction in cost in using the model versus random sampling. In contrast to other performance measures, RA is meaningful and relevant to experts in the domain.

A.2 Definition of RA

In the following, 'the model' refers to the learned recognition model for predicting whether a sequence is a NPP.

We define a *relative advantage* (RA) function which predicts the reduction in cost in using the model versus random sampling.

$$RA = \frac{A}{B} \tag{1}$$

where

A = the expected cost of finding one NPP by repeated independent random sampling from SWISS-PROT and performing a laboratory analysis of each protein.

B = the expected cost of finding one NPP by repeated independent random sampling from SWISS-PROT and analysing only those proteins which are predicted by the learned model to be a NPP.

RA can be defined in terms of probability as follows. Let

C = the cost of testing the biological activity of one protein via wet-experiments in the laboratory;

NPP = Sequence is a NPP;

Rec = Model recognises sequence as a NPP.

Equation 1 can now be rewritten as:

$$RA = \frac{C/Pr(NPP)}{C/Pr(NPP \mid Rec)} = \frac{Pr(NPP \mid Rec)}{Pr(NPP)} \tag{2}$$

Let testing the model on test data yield the 2×2 contingency table shown in Table 9 with the cells n_1, n_2, n_3, and n_4. Let $n = n_1 + n_2 + n_3 + n_4$ be the number of instances in the test-set. Note that the random set of sequences

Table 9: 2×2 Contingency table for the test-set. The axes of the 2×2 matrix are labelled by the sets NPP sequences, Random sequences, H (Hypothesis predictions) and $\overline{H}$ (complement of H). The cells of the matrix represent the cardinalities of the corresponding intersections of these sets. $n_1+n_2+n_3+n_4 = n$, where n is the number of instances in the test-set.

	Set of test NPP sequences	Set of test Random sequences
H	n_1	n_2
$\overline{H}$	n_3	n_4

Table 10: 2×2 Contingency table for the positives and negatives in the test-set. The axes of the 2×2 matrix are labelled by the sets NPP sequences, Negative sequences, H (Hypothesis predictions) and $\overline{H}$ (complement of H). The cells of the matrix represent the cardinalities of the corresponding intersections of these sets. $\delta = M/S$ where S is the total number of sequences in the entire SWISS-PROT database, of which M are NPPs.

	Set of test NPP sequences	Set of test Negative sequences
H	n_1	$n_2\ (1-\delta)$
$\overline{H}$	n_3	$n_4\ (1-\delta)$

referred to in the right-hand column may include some NPP sequences. Table 10 shows an estimate of the contingency table that would be obtained if it were possible to identify and remove all the positives from the set of randoms. If the proportion of NPPs in the test-set was known to be the same as the proportion of NPPs in the database then we could estimate $Pr(NPP)$ to be $(n_1+n_3)/n$ and $Pr(NPP \mid Rec)$ to be $n_1/(n_1+n_2)$. These estimates cannot be used with our method because we cannot assume that the proportion of NPPs is the same in the test-set and database.

In order to derive a formula for estimating RA given both a set of positives and a set of randoms, we estimate $Pr(NPP)$ and $Pr(NPP \mid Rec)$ as follows. Let S be the total number of sequences in the database, of which M are NPPs.

$$\begin{aligned} Pr(NPP) &= \frac{no.\ of\ NPPs\ in\ the\ database}{no.\ of\ sequences\ in\ the\ database} \\ &= M/S \end{aligned} \tag{3}$$

$$Pr(NPP \mid Rec) = \frac{N_{db\_NPP\_recog}}{N_{db\_seq\_pred\_pos}} \tag{4}$$

where $N_{db\_NPP\_recog}$ is the number of NPPs in db which are recognised by model and $N_{db\_seq\_pred\_pos}$ is the number of sequences in db which the model predicts to be NPP.

Table 11 shows the expected result of using the learned recognition model on the entire SWISS-PROT database. Note that the factor $(1-\delta)$ does not appear as it cancels out. From Equation 4 and Table 11 it follows that:

Table 11: 2×2 Contingency table for SWISS-PROT. The axes of the 2×2 matrix are labelled by the sets NPP sequences, Random sequences, H (Hypothesis predictions) and $\overline{H}$ (complement of H). The total of the counts/frequencies in the four cells $= S$, where S is the total number of sequences in the SWISS-PROT database.

	NPP sequences in SWISS-PROT	Negative sequences in SWISS-PROT
H	$\left(\frac{n_1}{n_1+n_3}\right) M$	$\left(\frac{n_2}{n_2+n_4}\right) (S-M)$
$\overline{H}$	$\left(\frac{n_3}{n_1+n_3}\right) M$	$\left(\frac{n_4}{n_2+n_4}\right) (S-M)$

$$\begin{aligned} Pr(NPP \mid Rec) &\simeq \frac{\left(\frac{n_1}{n_1+n_3}\right) \times M}{\left(\frac{n_1}{n_1+n_3}\right) M + \left(\frac{n_2}{n_2+n_4}\right) (S-M)} \\ &= (Mp_1)/(Mp_1 + (S-M)p_2) \end{aligned} \tag{5}$$

where $p_1 = n_1/(n_1+n_3)$ and $p_2 = n_2/(n_2+n_4)$. Substituting Equations 3 and 5 into Equation 2 gives

$$\begin{aligned} RA &= \frac{(Mp_1)/(Mp_1 + (S-M)p_2)}{M/S} \\ &= \frac{Sp_1}{Mp_1 + (S-M)p_2} \\ &= \frac{Sp_1}{Sp_2 + M(p_1 - p_2)} \end{aligned} \tag{6}$$

A.3 Estimating Relative Advantage

In the following Relative Advantage over the entire population is represented by RA in capital letters where as Relative Advantage over a sample is denoted by lower case i.e. ra. As the value of M is not known, we estimate $\sum_{M=57}^{90} RA$. Therefore we integrate Equation 6 with respect to M. The lower limit of M is equal to the number of known NPPs in SWISS-PROT. The upper limit of M is the most probable number of NPPs in SWISS-PROT i.e. a total of the known NPPs and those proteins which have yet to be scientifically recognised as a NPP.

$$\begin{aligned} \sum_{M=57}^{90} RA &\simeq Sp_1 \times \int_{M=57}^{91} \frac{1}{(p_1-p_2)M + Sp_2} \partial M \\ &= \left[\frac{Sp_1}{(p_1-p_2)} \ln((p_1-p_2)M + Sp_2) + k \right]_{57}^{91} \\ &= \frac{Sp_1}{(p_1-p_2)} \ln \frac{91(p_1-p_2) + Sp_2}{57(p_1-p_2) + Sp_2} \end{aligned} \tag{7}$$

We estimate $\sum_{M=57}^{90} RA$ by summing an estimate of the $\sum_{M=57}^{90} RA$ for each instance in the test-set as follows, where n is the number of instances in the test-set. This method has the advantage that it allows the significance of the difference between the RA of two models to be gauged (see Section A.4).

$$\sum_{k=1}^{n} \sum_{M=57}^{90} ra_k \tag{8}$$

From Equation 8 and the contingency table it follows that:

$$\sum_{M=57}^{90} ra = \frac{1}{n} \sum_{i=1}^{4} \left(n_i \sum_{M=57}^{90} ra_i \right) \tag{9}$$

Each $\sum_{M=57}^{90} ra_i$ is estimated by substituting $p_1 = \frac{a}{a+c}$ and $p_2 = \frac{b}{b+d}$ into Equation 7. The values of a, b, c and d are determined by three steps.

1. Whatever the i value, a, b, c and d are initially given the values of the corresponding counts/frequencies in the contingency table for the test-set (see Table 9).

2. Each one of a, b, c and d, is decremented providing that the value before subtraction is greater than 1.

   We do not decrement when the value before subtraction is zero because this can result in p_1 or p_2 having negative values; this does not make sense because p_1 and p_2 are probabilities. We do not decrement when the value is one because this can cause p_1 or p_2 to have the value zero, which in turn has a *highly disproportionate* effect on the value of $\sum_{M=57}^{90} ra_i$.

3. The value of either a, b, c or d is incremented to reflect the classification of an instance in the cell n_i.

For instance, if $i = 2$ and all the counts in the contingency table are greater than one then $a = n_1 - 1, b = n_2, c = n_3 - 1, d = n_4 - 1$.

Note that Steps 1 and 2 assign the same prior probability to each instance because the effect of each step is not dependent upon which cell the current instance belongs to. Therefore this method of estimating $\sum_{M=57}^{90} RA$ has the properties of a) producing identically distributed random variables representing the outcome for each instance; b) having a sample mean which approaches the population mean in the limit and c) having a relatively small sample variance.

The final step of our method for estimating RA is to take the mean of the summed values.

$$Mean\ RA = \frac{\sum_{M=57}^{90} ra_i}{90 - (57 - 1)} = \frac{\sum_{M=57}^{90} ra_i}{34} \tag{10}$$

A.4 Assessing the significance of the difference between the RA of two models

Next we develop a method for assessing the significance of the difference between the RA of two models. This method tackles a problem which is similar to that posed by the third question in Dietterich's taxonomy of statistical questions in Machine Learning [9]. That is, how to choose between classifiers for a single application domain in which the amount of available data is sufficient to allow some of it to be set aside for evaluating

Table 12: 4×4 Contingency Table. The rows of the 4×4 matrix are labelled by the cells of the 2×2 contingency table for H_1. The columns of the 4×4 matrix are labelled by the cells of the 2×2 contingency table for H_2. The cells of the 4×4 matrix represent the cardinalities of the corresponding intersections of these sets. $\sum_{i=1}^{4} \sum_{j=1}^{4} n_{i,j} = n$, where n is the number of instances in the test-set.

	n_1	n_2	n_3	n_4
n_1	$n_{1,1}$	$n_{1,2}$	$n_{1,3}$	$n_{1,4}$
n_2	$n_{2,1}$	$n_{2,2}$	$n_{2,3}$	$n_{2,4}$
n_3	$n_{3,1}$	$n_{3,2}$	$n_{3,3}$	$n_{3,4}$
n_4	$n_{4,1}$	$n_{4,2}$	$n_{4,3}$	$n_{4,4}$

classifiers. However a new method is needed because of the fundamental differences between Relative Advantage and predictive accuracy.

We compare the performance of two recognition models, H_1 and H_2, by comparing their $\sum_{M=57}^{90} RA$ values. Let d be difference in $\sum_{M=57}^{90} RA$ values over the entire population, i.e. for all the proteins in SWISS-PROT, and $\hat{d}$ be the observed difference on the test-set.

$$d = \sum_{M=57}^{90} RA_{H_1} - \sum_{M=57}^{90} RA_{H_2} \tag{11}$$

$$\hat{d} = \sum_{M=57}^{90} ra_{H_1} - \sum_{M=57}^{90} ra_{H_2} \tag{12}$$

$\hat{d}$ is an unbiased estimator for the true difference because it is calculated using an independent test-set. To determine whether the observed difference is statistically significant we address the following question. What is the probability that $\sum_{M=57}^{90} RA_{H_1} > \sum_{M=57}^{90} RA_{H_2}$, given the observed difference, $\hat{d}$.

If D is a random variable representing the outcome of estimating d by random sampling then, according to the Central Limit Theorem, $\hat{\mu}_D$ is normally distributed in the limit. It has an estimated mean $\hat{d}$ and has an estimated variance of $\hat{\sigma}_D^2/n$. The variance of a random variable, X, is $\sigma_X^2 = E((X)^2) - (E(X))^2$. Therefore, since D is a random variable:

$$\hat{\sigma}_D^2 = \hat{\mu}_{D^2} - \hat{\mu}_D^2 \tag{13}$$

We calculate $\hat{\mu}_{D^2}$ as follows. Let testing the model on test data yield the 4×4 contingency table shown in Table 12 with the cells $n_{i,j}$.

$$\hat{\mu}_{D^2} = \frac{1}{n} \sum_{i=1}^{4} \sum_{j=1}^{4} \left(n_{i,j} \left(\sum_{M=57}^{90} ra_i - \sum_{M=57}^{90} ra_j \right)^2 \right) \tag{14}$$

Given that $p(\sum_{M=57}^{90} RA_{H_1} > \sum_{M=57}^{90} RA_{H_2}) = p(\sum_{M=57}^{90} RA_{H_1} - \sum_{M=57}^{90} RA_{H_2} > 0)$ we evaluate our null hypothesis by estimating $p(d < 0)$ using the Central Limit Theorem.

$$\int_{x=-\infty}^{0} Pr(d = x)dx = \int_{x=-\infty}^{0} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx \tag{15}$$

where $\mu = \hat{\mu}_D$ and $\sigma = \hat{\sigma}_D/\sqrt{n}$.

B CProgol and the Production Rules Generated

Table 13 shows the production rules generated by CProgol. The rules comply with Prolog syntax. $signal(X, Y)$ is true if there is a signal peptide at the beginning of the sequence X, and it is followed by a sequence Y. The other dyadic predicates are defined similarly. Non-terminals and terminals which appear on the right hand side of the production rules listed in Table 13 are defined in Table 14. Table 14 shows the Prolog code representing the background knowledge input to Progol. The production rules, when taken together with the partial grammar shown in Figure 3, form a grammar for NPP sequences.

B.1 Defining a Hypothesis Language for Progol

A Hypothesis Language for Progol is defined by:–

- mode and type declarations which state the forms that atoms in hypotheses may take (see Sections B.1.1 and B.1.2);
- prune declarations which further restrict the form of hypotheses (see Section B.1.3);
- the maximum number of layers of variables introduced by atoms in the body of induced clauses from variables in the head of the clauses;
- the maximum number of literals in the body of induced clauses.

The hypothesis language used in the experiment is defined by Tables 15, 16 and 17.

B.1.1 Mode Declarations

The mode declarations state the mode of call for those predicates that can appear in a hypothesis induced by Progol. There are two types of mode declarations, as shown below. The first describes the form of literals that may appear in the head of clauses induced by Progol and the second describes those that may appear in the body.

```
:- modeh(Recall_number, Head_template)?
:- modeb(Recall_number, Body_literal_template)?
```

`Head_template` and `Body_literal_template` are templates of predicates and take the form `predicate(ts1, ts2, ... )`, where `ts` is a term specification. Each term specification comprises two parts: a mode and a type. Types are described in Section B.1.2. The three possible modes are:–

+ This indicates that the term is an input. That is, in all calls to this predicate, the term will be bound to a value.

– This indicates the term is an output.

\# This indicates that a constant should appear in this term.

`Recall_number` refers to the determinacy of the predicate template, that is it specifies the maximum number of times a call to the predicate can succeed for a given set of input variables. Hence for determinate predicate templates it is set to one and for indeterminate predicate templates to values greater than one. If the User specifies a `Recall_number` to be `*` then Progol assigns a default value of 100 to it.

B.1.2 Type Declarations

Types that are included in mode declarations may be unary predicates defined in the background knowledge. All but one of the mode declarations listed in Table 15 refer to the type `rlist/1`; this is a predicate whose definition is listed on Table 14. Progol type-checks a constant by executing a query in which the predicate corresponds to the type and the term is instantiated to the constant. If the query succeeds then Progol accepts that the constant is of the correct type.

B.1.3 Prune Declarations

Prune declarations are used to prevent Progol considering specified forms of clauses. A declaration is made by the User defining the predicate `prune(Head, Body)`. Progol will not consider a clause if a call to `prune(Head, Body)` succeeds when `Head` is instantiated to the literal in the head of the proposed clause and `Body` is instantiated to the proposed body of the clause.

B.2 The Time and Space Complexity of Progol

The Progol algorithm, as analysed in [20], has time and space complexity which increases linearly in both the number of examples and the number of clauses in the learned theory. However, the scaling constants involved vary depending on the size of the hypothesis space searched for each clause. This is controlled using a number of parameters, including a clause length bound and a proof depth bound.

Table 13: Production Rules Generated by CProgol

```
sigpep(A,B) :- g(A,C), star(C,D), s(D,B).
sigpep(A,B) :- m(A,C), star(C,D), hydrophilic(D,E), tiny(E,B).
sigpep(A,B) :- hydrophobic(A,C), star(C,D), w(D,E), hydro_b_acc(E,B).
sigpep(A,B) :- large(A,C), run(C,D,hydrophobic), star(D,E), t(E,B).
sigpep(A,B) :- m(A,C), star(C,D), t(D,E), neutral(E,F), small(F,B).
sigpep(A,B) :- m(A,C), star(C,D), very_hydrophobic(D,E), positive(E,F), tiny(F,B).
sigpep(A,B) :- hydrophobic(A,C), run(C,D,hydrophobic), star(D,E), f(E,F), hydrophobic(F,B).
sigpep(A,B) :- hydrophobic(A,C), star(C,D), h(D,E), hydrophobic(E,F), tiny(F,B).
sigpep(A,B) :- hydrophobic(A,C), star(C,D), v(D,E), hydrophobic(E,F), neutral(F,B).
sigpep(A,B) :- large(A,C), star(C,D), a(D,E), hydrophobic(E,F),small(F,B).
sigpep(A,B) :- large(A,C), star(C,D), s(D,E), neutral(E,F), small(F,B).

start(A,B) :- a(A,C), very_hydrophobic(C,B).
start(A,B) :- d(A,C), t(C,B).
start(A,B) :- g(A,C), v(C,B).
start(A,B) :- h(A,C), r(C,B).
start(A,B) :- k(A,C), r(C,B).
start(A,B) :- l(A,C), r(C,B).
start(A,B) :- q(A,C), g(C,B).
start(A,B) :- s(A,C), l(C,B).
start(A,B) :- w(A,C), q(C,B).
start(A,B) :- hydrophilic(A,C), a(C,B).
start(A,B) :- hydrophilic(A,C), hydrophilic(C,B).
start(A,B) :- positive(A,C), k(C,B).
start(A,B) :- small(A,C), r(C,B).

middle(A,B) :- yvh(A,C), star(C,D), large(D,E), large(E,B).
middle(A,B) :- positive(A,C), star(C,D), neutral(D,E), large(E,F), large(F,B).
middle(A,B) :- hydro_b_acc(A,C), star(C,D), hydrophobic(D,E),neutral(E,F), aromatic(F,B).
middle(A,B) :- hydro_b_acc(A,C), yvh(C,D), star(D,B).
middle(A,B) :- small(A,C), star(C,D), p(D,E), large(E,F), large(F,B).
middle(A,B) :- y(A,C), star(C,D), g(D,E), hydrophobic(E,B).
middle(A,B) :- hydro_b_acc(A,C), star(C,D), k(D,E), neutral(E,F), small(F,B).
middle(A,B) :- small(A,C), star(C,D), l(D,E), m(E,B).
middle(A,B) :- small(A,C), star(C,D), f(D,E), hydrophobic(E,F),aliphatic(F,B).
middle(A,B) :- tiny(A,C), star(C,D), m(D,B).
middle(A,B) :- q(A,C), star(C,D), positive(D,E), neutral(E,F),neutral(F,B).
middle(A,B) :- hydrophobic(A,C), star(C,D), m(D,E), hydrophilic(E,F), neutral(F,B).
middle(A,B) :- e(A,C), star(C,D), i(D,B).
middle(A,B) :- q(A,C), star(C,D), l(D,B).
middle(A,B) :- aromatic(A,C), star(C,D), v(D,E), neutral(E,F),hydro_b_don(F,B).
middle(A,B) :- aromatic(A,C), star(C,D), a(D,E), e(E,B).
middle(A,B) :- c(A,C), star(C,D), c(D,B).
middle(A,B) :- y(A,C), star(C,D), hydro_b_don(D,E), hydro_b_don(E,B).
middle(A,B) :- hmn(A,C), star(C,D), d(D,B).
middle(A,B) :- tiny(A,C), star(C,D), l(D,E), hydro_b_don(E,F),hydro_b_don(F,B).
middle(A,B) :- neutral(A,C), star(C,D), very_hydrophobic(D,E),negative(E,F), aromatic(F,B).
middle(A,B) :- h(A,C), star(C,D), very_hydrophobic(D,E), neutral(E,B).
middle(A,B) :- h(A,C), star(C,D), positive(D,E), neutral(E,F),hydro_b_don(F,B).
middle(A,B) :- hydrophilic(A,C), star(C,D), e(D,E), small(E,B).
middle(A,B) :- hydro_b_don(A,C), star(C,D), g(D,E), hydrophobic(E,F), neutral(F,B).
middle(A,B) :- hydrophobic(A,C), star(C,D), n(D,E), neutral(E,F), large(F,B).
middle(A,B) :- hydrophobic(A,C), star(C,D), a(D,E), f(E,B).
middle(A,B) :- hydro_b_don(A,C), star(C,D), negative(D,E), aromatic(E,B).
middle(A,B) :- hydro_b_acc(A,C), star(C,D), r(D,E), hydrophobic(E,B).
middle(A,B) :- aromatic(A,C), star(C,D), a(D,E), very_hydrophobic(E,F), large(F,B).
middle(A,B) :- tiny(A,C), star(C,D), r(D,E), tiny(E,B).

end(A,B) :- pp(A,C), d(C,B).
end(A,B) :- pp(A,C), large(C,B).
end(A,B) :- e(A,C), l(C,D), s(D,B).
end(A,B) :- e(A,C), v(C,D), v(D,B).
end(A,B) :- g(A,C), positive(C,D),
            hydro_b_don(D,B).
end(A,B) :- q(A,C), a(C,D), g(D,B).
end(A,B) :- r(A,C), tiny(C,D), hydro_b_acc(D,B).
end(A,B) :- t(A,C), neutral(C,D), hydro_b_acc(D,B).
end(A,B) :- positive(A,C), r(C,D), small(D,B).
end(A,B) :- positive(A,C), r(C,D), hydro_b_acc(D,B).
end(A,B) :- large(A,C), l(C,D), v(D,B).
end(A,B) :- small(A,C), hydrophobic(C,D), positive(D,B).
end(A,B) :- tiny(A,C), star(C,D), r(D,B).
end(A,B) :- aliphatic(A,C), n(C,D), t(D,B).
```

Table 14: Background Knowledge Predicates. The ground instantiations of the unary predicates representing the properties shown in Table 3 are not shown here for reasons of space.

```
rlist([]).
rlist([R|T]) :- res(R), rlist(T).
res(a).  res(b).  res(c).   ...   res(z).

any([_|S],S). % residue of any type or property

kr(A,C) :- k(A,B), r(B,C).                           kp(A,C) :- k(A,B), positive(B,C).
pp(A,C) :- positive(A,B), positive(B,C).             gkr(A,D) :- g(A,B), k(B,C), r(C,D).
grr(A,D) :- g(A,B), r(B,C), r(C,D).                  yvh(A,C) :- y(A,B), very_hydrophobic(B,C).
hp(B,C) :- h(B,C), p(C,D).
hmn(A,E) :- hydrophilic(A,B), star(B,C), m(C,D), negative(D,E).
wmdf(A,B) :- w(A,C), m(C,D), d(E,F), f(F,B), end(B).
very_hydrophobic([R|T],T):- very_hydrophobic(R).  small([R|T],T):- small(R).
hydrophobic([R|T],T):- hydrophobic(R).            tiny([R|T],T):- tiny(R).
hydrophilic([R|T],T):- hydrophilic(R).            polar([R|T],T):- polar(R).
positive([R|T],T):- positive(R).                  aliphatic([R|T],T):- aliphatic(R).
negative([R|T],T):- negative(R).                  aromatic([R|T],T):- aromatic(R).
neutral([R|T],T):- neutral(R).                    hydro_b_don([R|T],T):- hydro_b_don(R).
large([R|T],T):- large(R).                        hydro_b_acc([R|T],T):- hydro_b_acc(R).

star(S,S).
star([_|S],T) :- star(S,T).

a([a|T],T). b([b|T],T). c([c|T],T).  ...   z([z|T],T).

run([X|S],T,P) :- prop(P), docall(P,X), run(S,T,P).
run([X,Y|S],S,P) :- prop(P), docall(P,X), docall(P,Y).
docall(P,X) :- Call=..  [P,X], Call.
prop(very_hydrophobic).  prop(hydrophobic).  prop(hydrophilic).
```

Table 15: Mode Declarations Used During Training with CProgol. TARGET_PREDICATE is either sigpep, start, middle or end.

```
:- modeh(1,TARGET_PREDICATE(+rlist,-rlist))?

:- modeb(1,yvh(+rlist,-rlist))?       :- modeb(*,hmn(+rlist,-rlist))?
:- modeb(1,hp(+rlist,-rlist))?        :- modeb(1,wmdf(+rlist,-rlist))?

:- modeb(1,a(+rlist,-rlist))?  :- modeb(1,b(+rlist,-rlist))?  ... :- modeb(1,z(+rlist,-rlist))?

:- modeb(1,hydrophobic(+rlist,-rlist))?        :- modeb(1,small(+rlist,-rlist))?
:- modeb(1,very_hydrophobic(+rlist,-rlist))?   :- modeb(1,tiny(+rlist,-rlist))?
:- modeb(1,hydrophilic(+rlist,-rlist))?        :- modeb(1,tiny(+rlist,-rlist))?
:- modeb(1,positive(+rlist,-rlist))?           :- modeb(1,aliphatic(+rlist,-rlist))?
:- modeb(1,negative(+rlist,-rlist))?           :- modeb(1,aromatic(+rlist,-rlist))?
:- modeb(1,neutral(+rlist,-rlist))?            :- modeb(1,hydro_b_don(+rlist,-rlist))?
:- modeb(1,large(+rlist,-rlist))?              :- modeb(1,hydro_b_acc(+rlist,-rlist))?

% The next five mode declarations were only used when generating rules for the ends.

:- modeb(1,pp(+rlist,-rlist))?         :- modeb(1,gkr(+rlist,-rlist))?
:- modeb(1,kp(+rlist,-rlist))?         :- modeb(*,run(+rlist,-rlist,#prop))?
:- modeb(1,kr(+rlist,-rlist))?

% The next mode declaration was only used when generating rules for signals, middles and ends.

:- modeb(*,star(+rlist,-rlist))?
```

Table 16: Prune Predicates Used During Training with CProgol. TARGET_PREDICATE is either sigpep, start, middle or end.

```
prune(_,Body):- in(star(A,B),Body), A==B.   % No star(X,X) in body

prune(Head,Body):- Head=.. [_,U,_], not(chain(U,Body)). % Body must form variable chain from head

prune(_,Body):- suffix(Body,Suffix),               % No star(X,Y),star(Y,Z) in body
                (Suffix=(star(_,_),(star(_,_),_))
                 ; Suffix=(star(_,_),(star(_,_))) ).

% The following prune was not used when generating the rules for the starts or middles.

prune(_,Body):- suffix(Body,Suffix),               % No run(X,Y),run(Y,Z) in body
                ( Suffix=(run(_,_,P),(run(_,_,P),_))
                 ; Suffix=(run(_,_,P),(run(_,_,P))) ).

% The following prune was not used when generating the start rules.

prune(_,star(_,_)).

:- TARGET_PREDICATE(x,y).     % Not allowed everything a NPP

chain(U,true).
chain(U,A):- A=.. [_,V,_|_], U==V.
chain(U,(A,B)):- A=.. [_,V,W|_], U==V, chain(W,B).

suffix(S,S).
suffix((_,S),S1):- suffix(S,S1).
```

Table 17: CProgol Settings Used for Training

	pos	inflate	i	c	nodes	v	h	r	s
signal	yes	100000	6	5	4000	0	100000000	100000000	100000000
start	yes	100000	6	5	4000	0	100000000	100000000	100000000
middle	yes	100000	6	5	1000	0	200	400	100000000
end	yes	100000	3	3	4000	0	100000000	100000000	100000000

pos The posonly setting. When this is set to **yes** CProgol adopts the ILP Bayesian approach to learning from positive examples.
inflate Controls the specificity of clauses obtained.
i An upper bound on the number of layers of variables introduced by atoms in the body of induced clauses from variables in the head of the clauses.
c An upper bound on the number of literals in the body of induced clauses.
nodes An upper bound on the nodes to be explored by CProgol when searching for a consistent clause.
v The verbosity of the output.
h A depth bound on the theorem prover.
r An upper bound on the number of resolutions beyond which the whole proof fails i.e. backtracking does *not* occur.
s Size of the unification stack in bytes.

C Rule-sets generated by C4.5 and C4.5 rules

The rule-set that was generated from the 'Proportions + Length + SIGNALP + Grammar' amalgam is shown in Figures 5 and 6. Each box contains a rule as it was output by C4.5rules together with the English translation which is shown in italics. The percentage in square brackets refers to the predicted accuracy of the corresponding rule. Each column of rules is tried in turn. Within each column, each rule is tried in order of appearance . The last rule is the 'default' rule which is used if none of the other rules apply.

The rule-set that was generated from the 'Proportions + Length' amalgam is shown in Figures 7, 8 and 9.

Rule 1:
`sigp_ymax <= 0.457`
`-> class 0 [99.9%]`
A sequence is not a NPP if the maximum Y score reported by SIGNALP ≤ 0.457.

Rule 31:
`length > 267`
`proportion_l <= 0.141717`
`proportion_polar <= 0.285366`
`-> class 0 [99.9%]`
A sequence is not a NPP if it is more than 267 residues long and the proportion of its residues which are:- 1) leucine (L) is ≤ 0.141717 *2) polar is* ≤ 0.285366.

Rule 14:
`gram_np_start_first = 0`
`proportion_h > 0.00588235`
`proportion_i > 0.0141343`
`proportion_neutral <= 0.793103`
`proportion_tiny > 0.208253`
`-> class 0 [99.9%]`
A sequence is not a NPP if the grammar predicts that a neuropeptide start is not present and the proportion of residues in the sequence which are: 1) histidine (H) is > 0.00588235; *2) isoleucine (I) is* > 0.0141343; *3) not surrounded by an electrostatic charge is* ≤ 0.793103; *4) tiny is* > 0.208253.

Rule 29:
`sigp_cmaxpos > 29`
`proportion_r > 0.047043`
`-> class 0 [99.8%]`
A sequence is not a NPP if the position where maximum SIGNALP C score is achieved is > 29 *and the proportion of its residues which are Arginine (R) is* > 0.047043

Rule 11:
`proportion_g > 0.040555`
`proportion_r > 0.047043`
`proportion_hydrophobic > 0.584906`
`proportion_tiny <= 0.208253`
`-> class 0 [99.7%]`
A sequence is not a NPP if the proportion of its residues which are:- 1) glycine (G) is > 0.040555; *2) arginine (R) is* > 0.047043; *3) hydrophobic is* > 0.584906; *4) tiny is* ≤ 0.208253.

Rule 27:
`sigp_cmaxpos <= 29`
`proportion_hydrophobic > 0.636591`
`proportion_tiny <= 0.301205`
`-> class 0 [99.7%]`
A sequence is not a NPP if the position where maximum SIGNALP C score is achieved is ≤ 29 *and the proportion of its residues which are:- 1) hydrophobic is* > 0.636591; *2) tiny* ≤ 0.301205.

Rule 5:
`proportion_tiny <= 0.176282`
`-> class 0 [99.7%]`
A sequence is not a NPP if the proportion of its residues which are tiny is ≤ 0.176282.

Rule 4:
`length <= 267`
`proportion_r <= 0.047043`
`-> class 0 [99.6%]`
A sequence is not a NPP if the number of residues in the sequence is ≤ 267 *and the proportion of its residues which are arginine (R) is* ≤ 0.047043.

Rule 9:
`gram_sig_l <= 25`
`proportion_g > 0.040555`
`proportion_q > 0.0395683`
`proportion_r > 0.047043`
`proportion_tiny <= 0.208253`
`-> class 0 [99.6%]`
A sequence is not a NPP if the grammar predicts that the length of the signal peptide is ≤ 25 *and the proportion of its residues which are:- 1) glycine (G) is* > 0.040555; *2) glutamine (Q) is* > 0.0395683; *3) arginine (R) is* > 0.047043; *4) tiny is* ≤ 0.208253.

Rule 26:
`proportion_i > 0.00882353`
`proportion_hydrophobic <= 0.636591`
`proportion_neutral > 0.793103`
`-> class 0 [99.5%]`
A sequence is not a NPP if the proportion of its residues which are:- 1) isoleucine (I) is > 0.00882353; *2) hydrophobic is* ≤ 0.636591; *3) not surrounded by an electrostatic charge is* > 0.793103.

Rule 6:
`gram_first <= 55`
`proportion_g <= 0.040555`
`-> class 0 [99.5%]`
A sequence is not a NPP if the grammar predicts that position of the first residue in the neuropeptide is less than or equal to 55 residues from the N-terminal and the proportion of residues in the sequence which are glycine is ≤ 0.040555.

Rule 23:
`gram_sig_l <= 27`
`proportion_p > 0.0873016`
`proportion_neutral <= 0.793103`
`-> class 0 [99.4%]`
A sequence is not a NPP if the grammar predicts that the length of the signal peptide is ≤ 27 *and the proportion of residues in the sequence which are:- 1) proline (P) is* > 0.0873016; *2) not surrounded by an electrostatic charge is* ≤ 0.793103.

Rule 33:
`length > 267`
`proportion_l > 0.142268`
`-> class 0 [98.4%]`
A sequence is not a NPP if it is more than 267 residues long and the proportion of its residues which are leucine (L) is > 0.142268.

Rule 17:
`gram_np_end_first = small`
`proportion_very_hydrophobic > 0.435146`
`-> class 0 [96.8%]`
A sequence is not a NPP if the grammar predicts that the third from last residue in the neuropeptide is small and the proportion of residues in the sequence which are very hydrophobic is > 0.435146.

Rule 21:
`gram_np_start_first = s`
`-> class 0 [95.8%]`
A sequence is not a NPP if the grammar predicts that the first residue in the neuropeptide is serine (S).

Figure 5: Rule-set generated from grammar amalgam.

```
Rule 20:
gram_np_start_first = q
-> class 0 [75.8%]
```
A sequence is not a NPP if the grammar predicts that the first residue in the neuropeptide is glutamine (Q).

```
Rule 36:
sigp_ymax > 0.457
proportion_e > 0.160121
-> class 1 [99.3%]
```
A sequence is a NPP if the maximum Y score reported by SIGNALP is $>$ 0.457 *and the proportion of its residues which are glutamic acid (E) is* $>$ 0.160121.

```
Rule 8:
sigp_ymax > 0.457
length <= 267
proportion_g > 0.040555
proportion_q <= 0.0395683
proportion_hydrophobic <= 0.584906
proportion_neutral <= 0.793103
proportion_tiny <= 0.208253
-> class 1 [99.2%]
```
A sequence is a NPP if the maximum Y score reported by SIGNALP is $>$ 0.457, *the sequence length is* $\leq$ 267, *the proportion of its residues which are:- 1) glycine (G) is* $>$ 0.040555; *2) glutamine (Q) is* $\leq$ 0.0395683; *3) hydrophobic is* $\leq$ 0.584906; *4) not surrounded by an electrostatic charge is* $\leq$ 0.793103; *5) tiny is* $\leq$ 0.208253.

```
Rule 24:
gram_sig_l > 27
sigp_cmaxpos <= 29
sigp_ymax > 0.457
proportion_hydrophobic <= 0.636591
proportion_neutral <= 0.793103
proportion_tiny > 0.208253
-> class 1 [99.2%]
```
A sequence is a NPP if the grammar predicts that the length of the signal peptide is $>$ 27, *the position where maximum SIGNALP C score is achieved is* $\leq$ 29, *the maximum Y score reported by SIGNALP is* $>$ 0.457 *and the proportion of residues in the sequence which are:- 1) hydrophobic* $\leq$ 0.636591; *2) not surrounded by an electrostatic charge is* $\leq$ 0.793103; *3) tiny is* $>$ 0.208253.

```
Rule 22:
sigp_cmaxpos <= 29
sigp_ymax > 0.457
length <= 267
proportion_g > 0.040555
proportion_p <= 0.0873016
proportion_r > 0.047043
proportion_hydrophobic <= 0.636591
proportion_neutral <= 0.793103
proportion_tiny > 0.208253
-> class 1 [98.9%]
```
A sequence is a NPP if the position where maximum SIGNALP C score is achieved is $\leq$ 29, *the maximum Y score reported by SIGNALP is* $>$ 0.457, *the number of residues in the sequence is* $\leq$ 267, *the proportion of residues in the sequence which are:- 1) glycine (G) is* $>$ 0.040555; *2) proline (P) is* $\leq$ 0.0873016; *3) arginine (R) is* $>$ 0.047043; *4) hydrophobic is* $\leq$ 0.636591; *5) not surrounded by an electrostatic charge is* $\leq$ 0.793103; *6) tiny is* $>$ 0.208253.

```
Rule 28:
sigp_ymax > 0.457
proportion_r > 0.047043
proportion_hydrophobic > 0.636591
proportion_tiny > 0.301205
-> class 1 [98.4%]
```
A sequence is a NPP if the maximum Y score reported by SIGNALP is $>$ 0.457, *the proportion of residues in the sequence which are arginine (R) is* $>$ 0.047043, *the proportion of residues in the sequence which are hydrophobic is* $>$ 0.636591 *and the proportion of residues in the sequence which are tiny is* $>$ 0.301205.

```
Rule 32:
sigp_ymax > 0.457
length > 267
proportion_l > 0.141717
proportion_l <= 0.142268
-> class 1 [98.3%]
```
A sequence is a NPP if the maximum Y score reported by SIGNALP is $>$ 0.457, *the number of residues in the sequence is* $>$ 267 *and the proportion of residues in the sequence which are leucine (L) is* $>$ 0.141717 *but* $\leq$ 0.142268.

```
Rule 34:
sigp_ymax > 0.457
sigp_smaxpos <= 5
length > 267
proportion_polar > 0.585366
-> class 1 [98.3%]
```
A sequence is a NPP if SIGNALP reports that the maximum Y score is $>$ 0.457 *and the position where maximum S score is achieved is* $\leq$ 5, *the number of residues in the sequence is* $>$ 267 *and the proportion of its residues which are polar is* $>$ 0.585366.

```
Rule 2:
gram_sig_l > 24
proportion_small <= 0.373913
-> class 1 [98.3%]
```
A sequence is a NPP if the grammar predicts that the length of the signal peptide is $>$ 24 *and the proportion of residues in the sequence which are small is* $\leq$ 0.373913.

```
Rule 3:
proportion_l > 0.134529
proportion_p > 0.0848656
proportion_t <= 0.0305344
-> class 1 [97.8%]
```
A sequence is a NPP if the proportion of its residues which are:- 1) leucine (L) is $>$ 0.134529; *2) proline (P) is* $>$ 0.0848656; *3) threonine (T) is* $\leq$ 0.0305344.

```
Default class: 0
```

Figure 6: Rule-set generated from grammar amalgam (*continued*).

```
Rule 28:
length > 267
proportion_r > 0.0350515
proportion_hydrophobic > 0.47046
-> class 0 [99.9%]
```
A sequence is not a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) arginine (R) is $>$ 0.0350515; *2) hydrophobic is* $>$ 0.47046.

```
Rule 25:
length > 267
proportion_q <= 0.0557491
proportion_hydrophobic > 0.47046
-> class 0 [99.9%]
```
A sequence is not a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) glutamine (Q) $\leq$ 0.0557491; *2) hydrophobic is* $>$ 0.47046.

```
Rule 16:
proportion_a <= 0.0870787
proportion_l > 0.0663812
proportion_m > 0.00917431
proportion_t > 0.0515464
-> class 0 [99.9%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) alanine (A) is $\leq$ 0.0870787; *2) leucine (L) is* $>$ 0.0663812; *3) methionine (M) is* $>$ 0.00917431; *4) threonine (T) is* 0.0515464.

```
Rule 12:
length > 185
proportion_c <= 0.0215054
proportion_aliphatic <= 0.238095
proportion_hydro_b_acc <= 0.427996
-> class 0 [99.9%]
```
A sequence is not a NPP if it contains more than 185 residues and the proportion of its residues which are:- 1) cysteine (C) is $\leq$ 0.0215054; *2) aliphatic is* $\leq$ 0.238095; *3) hydrogen bond acceptors is* $\leq$ 0.427996 .

```
Rule 18:
proportion_t > 0.0515464
proportion_very_hydrophobic >
0.374718
proportion_aliphatic <= 0.238095
-> class 0 [99.8%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) threonine (T) is $>$ 0.0515464; *2) very hydrophobic is* $>$ 0.374718; *3) aliphatic is* $\leq$ 0.238095.

```
Rule 27:
length > 267
proportion_s > 0.0778443
proportion_hydrophobic > 0.47046
-> class 0 [99.8%]
```
A sequence is not a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) serine (S) is $>$ 0.0778443; *2) hydrophobic is* $>$ 0.47046.

```
Rule 11:
length > 185
proportion_k <= 0.0705882
proportion_t <= 0.0515464
proportion_hydro_b_acc <= 0.427996
-> class 0 [99.8%]
```
A sequence is not a NPP if it contains more than 185 residues and the proportion of its residues which are:- 1) lysine (K) is $\leq$ 0.0705882; *2) threonine (T)* $\leq$ 0.0515464; *3) hydrogen bond acceptors is* $\leq$ 0.427996.

```
Rule 7:
proportion_a <= 0.0818386
proportion_aromatic > 0.133641
-> class 0 [99.7%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) alanine (A) is $\leq$ 0.0818386; *2) aromatic is* $>$ 0.133641.

```
Rule 3:
proportion_tiny <= 0.176282
-> class 0 [99.7%]
```
A sequence is not a NPP if the proportion of its residues which are tiny is $\leq$ 0.176282.

```
Rule 1:
length <= 267
proportion_r <= 0.0472245
-> class 0 [99.6%]
```
A sequence is not a NPP if it contains fewer than 268 residues and the proportion of its residues which are arginine (R) is $\leq$ 0.0472245.

```
Rule 2:
proportion_s <= 0.0514019
-> class 0 [99.6%]
```
A sequence is not a NPP if the proportion of its residues which are serine (S) is $\leq$ 0.0514019.

```
Rule 20:
proportion_r > 0.0472245
proportion_negative > 0.044586
proportion_aliphatic > 0.238095
-> class 0 [99.5%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) arginine (R) is $>$ 0.0472245; *2) negative is* $>$ 0.044586; *3) aliphatic is* $>$ 0.238095.

```
Rule 4:
proportion_d <= 0.0222222
-> class 0 [99.3%]
```
A sequence is not a NPP if the proportion of its residues which are aspartic acid is $\leq$ 0.0222222.

```
Rule 5:
proportion_l <= 0.0663812
proportion_hydro_b_acc <= 0.427996
-> class 0 [99.3%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) leucine (L) is $\leq$ 0.0663812; *2) hydrogen bond acceptors is* $\leq$ 0.427996.

```
Rule 21:
length <= 267
proportion_hydro_b_acc > 0.427996
-> class 0 [99.3%]
```
A sequence is not a NPP if it contains fewer than 268 residues and the proportion of its residues which are hydrogen bond acceptors is $>$ 0.427996.

```
Rule 9:
length <= 185
proportion_c > 0.0247253
proportion_c <= 0.1
proportion_t <= 0.0515464
-> class 0 [98.5%]
```
A sequence is not a NPP if it contains fewer than 185 residues and the proportion of its residues which are:- 1) cysteine (C) is $>$ 0.0247253 *but* $\leq$ 0.1; *2) threonine (T) is* $\leq$ 0.0515464.

```
Rule 22:
proportion_m <= 0.0177936
proportion_w <= 0.0139276
proportion_hydrophobic <= 0.47046
-> class 0 [91.2%]
```
A sequence is not a NPP if the proportion of its residues which are:- 1) methionine (M) is $\leq$ 0.0177936; *2) tryptophan (W)* $\leq$ 0.0139276; *3) hydrophobic is* $\leq$ 0.47046.

Figure 7: Rule-set generated from 'Proportions + Length' amalgam.

```
Rule 13:
length > 185
length <= 267
proportion_c > 0.0215054
proportion_k > 0.0705882
proportion_l > 0.0663812
proportion_r > 0.0472245
proportion_s > 0.0514019
proportion_t <= 0.0515464
proportion_aliphatic <= 0.238095
-> class 1 [99.6%]
```

A sequence is a NPP if the proportion of its residues which are:- 1) cysteine (C) is > 0.0215054*; 2) lysine (K) is* > 0.0705882*; 3) leucine (L) is* > 0.0663812*; 4) arginine (R) is* > 0.0472245*; 5) serine (S) is* > 0.0514019*; 6) threonine (T) is* ≤ 0.0515464*; 7) aliphatic is* ≤ 0.238095.

```
Rule 6:
length <= 185
proportion_c <= 0.0247253
proportion_d > 0.0222222
proportion_l > 0.0663812
proportion_r > 0.0472245
proportion_s > 0.0514019
proportion_t <= 0.0515464
proportion_tiny > 0.176282
proportion_aliphatic <= 0.238095
proportion_aromatic <= 0.133641
proportion_hydro_b_acc <= 0.427996
-> class 1 [99.4%]
```

A sequence is a NPP if it contains fewer than 186 residues and the proportion of its residues which are:- 1) cysteine (C) is ≤ 0.0247253*; 2) aspartic acid (D) is* > 0.0222222*; 3) leucine (L) is* > 0.0663812*; 4) arginine (R) is* > 0.0472245*; 5) serine (S) is* > 0.0514019*; 6) threonine (T) is* ≤ 0.0515464*; 7) tiny is* > 0.176282*; 8) aliphatic is* ≤ 0.238095*; 9) aromatic is* ≤ 0.133641*; 10) hydrogen bond acceptor is* ≤ 0.427996.

```
Rule 8:
length <= 185
proportion_a > 0.0818386
proportion_c <= 0.0247253
proportion_d > 0.0222222
proportion_l > 0.0663812
proportion_r > 0.0472245
proportion_s > 0.0514019
proportion_t <= 0.0515464
proportion_aliphatic <= 0.238095
proportion_hydro_b_acc <= 0.427996
-> class 1 [99.4%]
```

A sequence is a NPP if it contains fewer than 186 residues and the proportion of its residues which are:- 1) alanine (A) is > 0.0818386*; 2) cysteine (C) is* ≤ 0.0247253*; 3) aspartic acid (D) is* > 0.0222222*; 4) leucine (L) is* > 0.0663812*; 5) arginine (R) is* > 0.0472245*; 6) serine (S) is* > 0.0514019*; 7) threonine (T) is* ≤ 0.0515464*; 8) aliphatic is* ≤ 0.238095*; 10) hydrogen bond acceptor is* ≤ 0.427996.

```
Rule 15:
length <= 267
proportion_d > 0.0222222
proportion_m <= 0.00917431
proportion_r > 0.0472245
proportion_s > 0.0514019
proportion_t > 0.0515464
proportion_very_hydrophobic <=
0.374718
proportion_tiny > 0.176282
proportion_hydro_b_acc <= 0.427996
-> class 1 [99.3%]
```

A sequence is a NPP if it contains fewer than 268 residues and the proportion of its residues which are:- 1) aspartic acid (D) is > 0.0222222*; 2) methionine (M) is* ≤ 0.00917431*; 3) arginine (R) is* > 0.0472245*; 4) serine (S) is* > 0.0514019*; 5) threonine (T) is* > 0.0515464*; 6) very hydrophobic is* ≤ 0.374718*; 7) tiny is* > 0.176282*; 8) hydrogen bond acceptor is* ≤ 0.427996.

```
Rule 17:
length <= 267
proportion_a > 0.0870787
proportion_l > 0.0663812
proportion_t > 0.0515464
proportion_very_hydrophobic <=
0.374718
proportion_tiny > 0.176282
proportion_hydro_b_acc <= 0.427996
-> class 1 [99.2%]
```

A sequence is a NPP if it contains fewer than 268 residues and the proportion of its residues which are:- 1) alanine (A) is > 0.0870787*; 2) leucine (L) is* > 0.0663812*; 3) threonine (T) is* > 0.0515464*; 4) very hydrophobic is* ≤ 0.374718*; 5) tiny is* > 0.176282*; 6) hydrogen bond acceptor is* ≤ 0.427996.

```
Rule 10:
length <= 267
proportion_c > 0.1
proportion_r > 0.0472245
proportion_tiny > 0.176282
proportion_hydro_b_acc <= 0.427996
-> class 1 [99.2%]
```

A sequence is a NPP if it contains fewer than 268 residues and the proportion of its residues which are:- 1) cysteine (C) is > 0.1*; 2) arginine (R) is* > 0.0472245*; 3) tiny is* > 0.176282*; 4) hydrogen bond acceptor is* ≤ 0.427996.

```
Rule 24:
length > 267
proportion_w > 0.0139276
proportion_hydrophobic <= 0.47046
-> class 1 [98.6%]
```

A sequence is a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) tryptophan (W) is > 0.0139276*; 4) hydrophobic is* ≤ 0.47046.

Figure 8: Rule-set generated from 'Proportions + Length' amalgam (*continued*).

```
Rule 23:
length > 267
proportion_m > 0.0177936
proportion_hydrophobic <= 0.47046
-> class 1 [98.4%]
```

A sequence is a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) methionine (M) is $>$ 0.0177936*; 2) hydrophobic is* $\leq$ 0.47046.

```
Rule 19:
length <= 267
proportion_d > 0.0222222
proportion_r > 0.0472245
proportion_s > 0.0514019
proportion_negative <= 0.044586
proportion_aliphatic > 0.238095
-> class 1 [98.3%]
```

A sequence is a NPP if it contains fewer than 268 residues and the proportion of its residues which are:- 1) aspartic acid (D) is $>$ 0.0222222; 2) arginine (R) is $>$ 0.0472245; 3) serine (S) is $>$ 0.0514019; 4) surrounded by a negative electrostatic charge is $\leq$ 0.044586; 5) aliphatic is $>$ 0.238095.

```
Rule 26:
length > 267
proportion_q > 0.0557491
proportion_r <= 0.0350515
proportion_s <= 0.0778443
proportion_hydrophobic > 0.47046
-> class 1 [96.3%]
```

A sequence is a NPP if it contains more than 267 residues and the proportion of its residues which are:- 1) glutamine (Q) $>$ 0.0557491*; 2) arginine (R) is* $\leq$ 0.0350515*; 3) serine (S) is* $\leq$ 0.0778443*; 4) hydrophobic is* $>$ 0.47046.

```
Default class:  0
```

Figure 9: Rule-set generated from 'Proportions + Length' amalgam (*continued*).

References

[1] N. Abe and H. Mamitsuka. Predicting protein secondary structure using stochastic tree grammars. *Machine Learning*, 29(2–3):275–301, 1997.

[2] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res*, 28:45–48, 2000.

[3] A. Bairoch, P. Bucher, and K. Hofman. The PROSITE database, its status in 1997. *Nucleic Acids Research*, 25:217–221, 1997.

[4] G.Y. Bakalkin, A.B. Rakhmaninova, V.K. Akparov, A.A. Volodin, V.V. Ovchinnikov, and R.A. Sarkisyan. Amino acid sequence pattern in the regulatory peptides. *International Journal of Peptide Protein Research*, 38:505–510, 1991.

[5] P. Baldi and S. Brunak. *Bioinformatics: the machine learning approach.* Cambridge, MA: MIT Press, 1998.

[6] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5:279–305, 1998.

[7] M.G. Claros, S. Brunak, and G. vonHeijne. Prediction of N-terminal sorting signals. *Current Opinion in Structural Biology*, 7(3):394–398, 1997.

[8] L. Devi. Consensus sequence for processing of peptide precursors at monobasic sites. *FEBS*, 280:189–194, 1991.

[9] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[10] R.F. Doolittle, editor. *Computer methods for macromolecular sequence analysis.* Academic Press, New York, 1996.

[11] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge: Cambridge University Press, 1998.

[12] S. Eddy. *HMMER User's Guide.* 4566 Scott Ave., St. Louis, MO 63110, USA, version 2.1.1 edition, December 1998.

[13] J.G. Henikoff and S. Henikoff. Blocks database and its applications. *Methods in Enzymology*, 266:88–105, 1996.

[14] L. Hillier, G. Lennon, M. Becker, F.M. Bonaldo, B. Chiapelli, S. Chissoe, N. Dietrich, T. DuBuque, A. Favello, W. Gish, M. Hawkins, M. Hultman, T. Kucaba, M. Lacy, M. Le, N. Le, E. Mardis, B. Moore, M. Morris, J. Parsons, C. Prange, L. Rifkin, T. Rohlfing, K. Schellenberg, M.B. Soares, F. Tan, J. Thierry-Mieg, E. Trevaskis, K. Underwood, P. Wohldman, R. Waterston, R. Wilson, and M. Marra. Generation and analysis of 280,000 human expressed sequence tags. *Genome Research*, 6:807–828, 1996.

[15] M.M. Klavdieva. The history of neuropeptides. *Frontiers in Neuroendocrinology*, 16:293–321, 1995.

[16] C. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 73–79, New York City, 1998.

[17] P. Linz. *An Introduction to Formal Languages and Automata.* D.C.Heath and Company, Lexington, Massachusetts, 1996.

[18] A. Lyall. Bioinformatics in the pharmaceutical industry. *Trends In Biotechnology*, 14:308–312, August 1996.

[19] R.S. Michalski and J. Wnek. Guest editors' introduction. *Machine Learning*, 27:205–208, June 1997.

[20] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[21] S. Muggleton. Inductive logic programming: issues, results and the LLL challenge. *Artificial Intelligence*, 114(1–2):283–296, December 1999.

[22] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[23] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[24] S.H. Muggleton. Learning from positive data. In S.H.Muggleton, editor, *Proceedings of the Sixth International Workshop on Inductive Logic Programming*, number 1314 in Lecture Notes in Artificial Intelligence, pages 358–376, Stockholm, Sweden, August 1996. Springer Verlag.

[25] S.H. Muggleton, C.H. Bryant, and A. Srinivasan. Learning chomsky-like grammars for biological sequence families. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 631–638, Stanford University, USA, 29 June - 2 July 2000. San Francisco, CA: Morgan Kaufmann.

[26] H Nielsen, S. Brunak, and G. vonHeijne. Machine learning approaches for the prediction of signal peptides and other protein sorting signals. *Protein Engineering*, 12(1):3–9, 1999.

[27] H. Nielsen, J. Engelbrecht, S. Brunak, and G. vonHeijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997.

[28] E. Pennisi. Human genome - Academic sequencers challenge Celera in sprint to the finish. *Science*, 283:1822–1823, 1999.

[29] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Artificial Intelligence*, pages 706–713, Menlo Park, CA, 1998. AAAI Press.

[30] J.R. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[31] C.J. Rawlings and D.B. Searls. Computational gene discovery and human disease. *Current Opinion in Genetic Development*, 7:416–423, 1997.

[32] M. Rholam, N. Brakch, D. Germain, D.Y. Thomas, C. Fahy, H. Boussetta, G. Boileau, and P. Cohen. Role of amino-acid-sequences flanking dibasic cleavage sites in precursor proteolytic processing – the importance of the first residue C-terminal of the cleavage site. *Euroupean Journal of Biochemistry*, 227:707–714, 1995.

[33] M. Rholam, P. Nicolas, and P. Cohen. Precursors for peptide-hormones share common secondary structures forming features at the proteolytic processing sites. *FEBS Letters,*, 207(1):1–6, 1986.

[34] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.

[35] D.B. Searls. The computational linguistics of biological sequences. In L. Hunter, editor, *Artificial Intelligence in Molecular Biology*, California, USA, 1993. AAAI Press.

[36] D.B. Searls. Linguistic approaches to biological sequences [review]. *Computer Applications in the Biosciences*, 13:333–344, August 1997.

[37] E.L. Sonnhammer, S.R. Eddy, E. Birney, A. Bateman, and R. Durbin. PFAM: multiple sequence alignments and HMM–profiles of protein domains. *Nucleic Acids Research*, 26:320–322, January 1998.

[38] E.L. Sonnhammer, S.R. Eddy, and R. Durbin. PFAM: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28:405–420, July 1997.

[39] P. Spence. Obtaining value from the human genome – a challenge for the pharmaceutical industry [review]. *Drug Discovery Today*, 3:179–188, April 1998.

[40] J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.