Sādhanā, Vol. 21, Part 3, June 1996, pp. 317–325. © Printed in India.

A machine learning system for identifying transmembrane domains from amino acid sequences

ARUL SIROMONEY[1] and RANI SIROMONEY[2]

[1]School of Computer Science and Engineering, Anna University, Madras 600 025, India
[2]Madras Christian College, Tambaram, Madras 600 059, India

Abstract. We present our machine learning system, that uses inductive logic programming techniques to learn how to identify transmembrane domains from amino acid sequences. Our system facilitates the use of operators such as 'contains', that act on entire sequences, rather than on individual elements of a sequence. The prediction accuracy of our new system is around 93%, and this compares favourably with earlier results.

Keywords. Machine learning; inductive logic programming; transmembrane domains; amino-acid sequences; molecular biology.

1. Introduction

A machine learning system that uses background knowledge relevant to the application area, and a set of examples and counter-examples of a particular concept, to learn the description of that concept in the form of a set of Horn clauses or Prolog program is an inductive logic programming system (Muggleton 1991).

We present in this paper, our implementation of a machine learning system, that uses inductive logic programming techniques to learn how to identify transmembrane domains from amino acid sequences. The problem of transmembrane identification is a very important protein classification problem. We present the work done by Shimozono *et al* (1993) in the identification of transmembrane domains, and then present the results obtained by using our machine learning system. Prediction accuracy by using our new system is found to be around 93%. These are very good results for any classification problem. These also compare very favourably with the earlier results that were of the same accuracy.

Our machine learning system extends earlier inductive logic programming techniques by facilitating the use of operators that act on entire sequences, rather than individual elements of a sequence. In this particular application, we use operators such as 'contains' and 'abstains' that are true when a test string is contained or not contained in a target string. However, these are not the only operators that can be used, and other operators that are useful in the biological context can be studied.

This paper describes the use of our new inductive logic programming technique to the identification of transmembrane domains. However, this technique is a general and extremely powerful tool that can be used to learn many biological and other concepts in sequence classification contexts.

2. Machine learning

Machine learning is the study and computer modelling of learning processes (Michalski *et al* 1983). Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation.

Machine learning systems can be classified along many different dimensions. One of the dimensions by which machine learning systems can be classified is on the basis of the underlying learning strategy used. There are many different learning strategies that can be used. The different strategies can be ordered by the amount of inference the learning system performs on the available information. Some strategies involve very little inference, whereas others entail a lot of inference.

In 'Rote Learning' or direct implanting of new knowledge, there is actually no inference or any transformation of the knowledge by the learner. The new knowledge is directly given to the learner and the learner uses this knowledge. 'Learning from Instruction' (or learning by being told) is the acquiring of knowledge from a teacher or some other organized source such as a textbook. The learner has to transform the knowledge from the input source to some internal representation. This has to be done in such a way that the new knowledge integrates well into any existing knowledge. In 'Learning by analogy', the learner uses existing knowledge that bears strong similarity to the desired new concept, and transforms or augments this knowledge to learn the new concept. In 'Learning from examples', given a set of examples and counter examples of a concept, the learner induces a general concept description that describes all of the positive examples and none of the negative examples. 'Learning from observation' includes discovery systems, theory-formation tasks, the creation of classification criteria to form taxonomic hierarchies, and other similar tasks that are to be done without the benefit of an external teacher. The learner is not provided with a set of instances of a particular concept, nor is it given access to some oracle that can classify any internally generated instances as positive or negative instances of the given concept.

The underlying learning strategy is only one of the dimensions along which machine learning systems can be classified. Machine learning systems can also be classified according to the type of knowledge acquired as well as according to the domain of the application.

Michalski (1983) describes inductive learning as that process of acquiring knowledge that is done by drawing inductive inferences from facts provided by either a teacher or the environment. Conceptual inductive learning is then described as a type of inductive learning whose final products are symbolic descriptions that are expressed in high-level human-oriented terms and forms. Concept learning from examples (which is also called

concept acquisition) is the task of inducing general descriptions of concepts from some specific instances of these concepts. These instances of the concepts are preclassified by a teacher into one or more classes (concepts). The induced description or hypothesis for a given concept is such that if an object satisfies this description, then it represents the concept.

Concept acquisition can learn a single concept or can learn a collection of concepts. In single concept learning, one can learn from positive examples alone, or from positive and negative examples (examples and counter-examples). When learning is from positive examples alone, since there are no counter-examples, there is no natural limit to which the description can be generalized. It is thus possible that the learnt description may be over-generalized. When counter-examples are also provided, there is an obvious limit on the extent to which the hypothesis can be generalized. The most useful counter-examples are the 'near-misses' that only slightly differ from positive examples. These clearly limit the extent to which the learnt hypothesis can be generalized.

Background knowledge defines the assumptions and constraints imposed on the observed facts and generated inductive assertions, and any relevant problem domain knowledge. In addition to describing the concept, the learnt hypothesis has to be consistent with the background knowledge.

3. Inductive logic programming

Muggleton (1991) describes inductive logic programming as the research area formed at the intersection of logic programming and machine learning, with work being done on problems of inductive reasoning within the confines of pure Prolog. Muggleton & Feng (1990) present the inductive logic programming learning algorithm GOLEM. Given a set of positive and negative examples, and background knowledge in the form of Horn clauses, GOLEM constructs a set of hypothesized Horn clauses which together cover all (or most) positive examples and no (or negligible) negative examples. GOLEM is implemented in 'C'. All the data is explicitly given as background knowledge.

Quinlan (1990, 1991) presents FOIL (first-order inductive learner) whose input is information about one or more relations. One of these relations is the target relation, and is to be defined by a Horn clause program. A set of tuples of constants that are in the target relation is given. A set of tuples that are not in the target relation may be given, or the closed world assumption used to generate such tuples. FOIL's output is a Horn clause program that defines the target relation.

FOIL and GOLEM belong to a group of learning algorithms that are described as covering methods. They construct a classification rule in the form of a disjunctive expression (a set of Horn clauses; or a Horn clause logic program). The covering method works as follows:

- A conjunction of conditions is found, such that it is satisfied by some examples in the target class, but no (or very few) examples outside the target class.
- This conjunction is appended as a disjunct of the classification rule.
- All examples that satisfy this conjunction are removed, and if there are still examples of the target class remaining to be covered, the procedure is repeated.

GOLEM determines each conjunction of conditions by taking the best conjunction (that covers the most positive examples, while covering less than a specified number of negative examples) of those formed from a random sample of pairs of examples, by choosing those background predicates that are true for both examples in the pair. The coverage of this best conjunction is further improved by using other examples in a random set of examples.

To sum up, inductive logic programming is a concept acquisition area in Machine Learning and the learning strategy used is that of 'Learning from examples'. Positive as well as negative examples are normally used. Background knowledge is also used. The knowledge acquired is in the form of a set of Horn clauses or a Prolog program.

4. Biological applications of inductive logic programming

The ILP algorithm GOLEM has been applied by Muggleton *et al* (1992) to the prediction of secondary structure from protein primary structure. This application predicts whether a particular residue is in an alpha-helix secondary structure or not. Positive examples are residue positions in the protein sequences that are in an alpha-helix secondary structure. Negative examples are those that are not. The background knowledge contains a lot of information about the protein structure, largely primary structure information. GOLEM is used to generate a Logic Program, that can predict whether a particular residue from a test protein sequence (that may not have been one of the training set used to learn this program), is in an alpha-helix secondary structure or not. Prediction accuracies of over 80% were achieved.

GOLEM is also used by King *et al* (1992) to model the quantitative structure-activity relationship (QSAR) in a related series of ligands. This is useful for drug design. Ranking of activities of drugs is done by considering pairs of drugs and comparing their activities. Thus the positive examples are paired examples of greater activity (where the first element of the pair is more active than the second element). Negative examples are paired examples of lower activity. Background facts are the chemical structure of the drugs and the properties of the substituents. GOLEM derived nine rules that predict the relative activities of two drugs.

GOLEM has not been applied to entire sequences. In the alpha-helix prediction, each element in the sequence is considered individually; each residue in the protein sequence is considered as an individual example or test case. However, many biological applications involve sequences and it becomes difficult to handle all the sequence applications by treating the elements of the sequence individually. There is a need for applying inductive logic programming techniques to entire sequences.

When the general description of a particular concept is learnt, there may be exceptions to the general description. Different sets of exceptions may each be specific to a particular general rule in the many that together, as a disjunct, describe the concept. These are local exceptions (Siromoney & Siromoney 1993). Local exceptions can be considered as a special case of the more general idea of variations (Siromoney & Siromoney 1995). A particular concept may be general in nature and of interest to many researchers. A lot of effort is spent in learning this concept. Another concept to be learnt may be similar to this, a variation of the other concept, and can be learnt with much less effort by making use of

the rules learnt to describe the earlier concept. The role of such variations is illustrated with a biological example. Identification of signal peptide sequences is learnt using mammalian data (of mammals other than rodents or primates). This knowledge is then used to easily learn the identification of signal peptides for primate data. It is also found in this illustrative experiment, that the prediction results were better than those obtained by learning from the primate data directly.

5. Identification of transmembrane domains

Arikawa *et al* (1992) and Shimozono *et al* (1993) describe a new approach to the identification of transmembrane domains from amino acid sequences. This problem of transmembrane identification is a very important protein classification problem. The PIR database contains amino acid sequences, with the FEATURE field for each sequence indicating where the transmembrane domains are located. The amino acid sequences are cut into substrings in such a manner that positive example strings contain substrings entirely within transmembrane domains, and negative example strings contain substrings that were completely outside the transmembrane domains. A decision tree is learnt that can classify any new substring as a transmembrane domain.

The simple form 'xAy' of a regular pattern language is used in the nodes of a decision tree. 'x' and 'y' are variable substrings and 'A' is a given fixed substring. Thus this simple form determines whether a given substring is 'contained' in the target substring. The 'contains' operator has been studied in detail by Sakakibara & Siromoney (1992). The 'contains' operator is true when the search string is contained in the target string. The 'Y' path of the node of the decision tree is taken when the target string is of the form 'xAy', that is 'A' is contained in the target string; and the 'N' path taken otherwise. The decision tree is learnt using a modification of the ID3 algorithm presented by Quinlan (1986).

When the amino acid sequences were used directly as the strings and the pattern, the performance was correct classification of 84.8% of the positive test cases and 89.6% of the negative test cases. The hydropathy index of an amino acid was used to distinguish the amino acids into three distinct categories. The twenty symbol amino acid sequences were transformed into three symbol sequences by assigning each amino acid symbol to one of these three distinct categories. The performance was then correct classification of 91.4% of the positive cases and 94.8% of the negative cases. A new method of indexing was used to transform again the twenty amino acid symbols to three symbols, and the performance was correct classification of 93.3% of the positive cases and 92.4% of the negative cases.

6. System description

Our new machine learning system uses the GOLEM algorithm. The major thrust of our new system is that the data need not be given explicitly as background knowledge. The background knowledge is in the form of C callable functions that operate on data corresponding to a particular example, and return a true or false value. This makes it much easier to specify the background knowledge, since the raw data is associated with each

example or test, and the background knowledge, in the form of callable functions, operates on this raw data and returns a true or false value. This also facilitates the use of operators such as 'contains'. The 'contains' operator returns true when the given search string is contained (is a substring) in the target string, and returns false otherwise. We also use the 'abstains' operator that is the opposite of 'contains' and is true when the search string is NOT contained in the target string.

Our machine learning system is written in 'C'. The inputs to our system are background knowledge, and positive and negative examples. The background knowledge is in the form of a C table, where each element specifies the address of the C function to be called (which will operate on the particular example and return a true or false value), and two parameters (currently!). The first parameter is a variable parameter, where the count and list of parameters is specified. The second parameter is a constant parameter. The use of the variable first parameter is equivalent to specifying the same function many times, each time with the corresponding parameter from the list of parameters. To illustrate: instead of specifying contains('++'), and contains('−−') as separate entries in the table of background knowledge, it is possible to specify them as a single entry that has the variable first parameter with a count of 2, and the list of parameters with "++" and "−−" as the two entries. The second constant parameter in this case is empty. This simplifies entry of background knowledge. The system can be easily extended to cater to more number of the constant parameters.

Each of the C functions used in this table also needs to be written and compiled along with this table. The parameters to any of these C functions include those that specify the particulars of the current example (or test case), and the two parameters as given in the table of background knowledge.

The examples are given as two text files, one containing all the positive examples and the other containing all the negative examples. Each example is the corresponding character sequence as a separate line in the text file.

The system uses the background knowledge by using each entry in the table. This involves repeatedly invoking the C function through its function address, and passing as parameters, the particulars of the current example (or test case), the current element in the list of first parameters and the constant second parameter. The function operates on the particular example, using the two parameters, and returns a true or false value. This is the equivalent of specifying the background knowledge through many ground Prolog clauses.

Our comprehensive system includes the machine learning system and also a test bed for using the learnt knowledge on a set of classified test cases. Background knowledge, positive and negative examples and also positive and negative test cases are given as input to our comprehensive system that first learns from the background knowledge and examples, and then predicts the test cases and measures the accuracy of prediction.

Our system currently uses the compiled table of C function addresses and the associated C functions as background knowledge rather than Prolog program clauses. It also currently internally evaluates the induced Horn clauses in their internal format rather than externally use a Prolog compiler on the Prolog output generated. Appropriate input/output translation modules can be added to use Prolog-like input and generate actual Prolog program output. The Prolog compiler will need a C function interface.

7. Method

Positive and negative examples of the transmembrane data from PIR were kindly supplied by Prof Miyano. These were sorted, only unique strings selected, and then randomized. There were 623 unique positive examples and 19164 unique negative examples. Three tests were conducted, each with 200 positive and 200 negative examples in the training set, and the remaining in the test set. The first, second and third 200 examples were taken for the three tests.

The background knowledge used modified versions of the contains and abstains operators which internally use one of two translation mechanisms on the raw amino-acid sequence of each example. Both the translation mechanisms converted the amino acid strings into three symbol strings. One was based on the hydropathy index of the amino acid, where the three categories of amino acids were based on the Kyte and Doolittle hydropathy index. The amino acids fall into three clear categories, with one category being amino acids with a positive hydropathy index, the second, those with very negative hydropathy index, and the third, those with hydropathy index near zero (0.0 to -1.6). This is the same translation mechanism used initially by Arikawa *et al* (1992). The second translation mechanism was based on whether the amino acid is acidic, basic or neutral. The variable first parameter was used to specify all the possible two and three character long sequences possible from the three symbols as the test strings for these operators ("++", "+*", "+−", and so on).

8. Results

The results of correct classification of the test set are given below.

		Correct +ve	Correct −ve
Test I	(Training: 1st 200)	92.7% (392)	93.5% (17725)
Test II	(Training: 2nd 200)	93.1% (394)	93.1% (17648)
Test III	(Training: 3rd 200)	93.6% (396)	93.2% (17671)

We present below one of the clauses that was learnt in the first test, as an example of the learnt rules:

```
contains('**'),  contains('+*'),  contains('+**'),
abstains('+-+'), abstains('+--'), abstains('-+-'),
contains('NN'),  contains('NNN'),
abstains('AB'),  abstains('AA'),  abstains('BBB'),
abstains('BBN'), abstains('BNA'), abstains('NBA'),
abstains('ANB'), abstains('ANA'),
```

where

* is an amino acid with positive hydropathy index
+ is an amino acid with 0 to −1.6 hydropathy index
− is an amino acid with a more negative hydropathy index

A is an acidic amino acid
B is a basic amino acid
N is a neutral amino acid

Redundant clauses such as

'abstains('AAA')' in 'abstains('AA'), abstains('AAA')'

were removed by hand to improve the readability of the learnt clause.

The results clearly indicate that our new machine learning system can be used to determine which amino-acid sequences fall within the transmembrane domain. The rules learnt by the system are however quite complex, as seen in the example given above, and require further analysis by human experts to see if any new biological results can be derived from them.

9. Conclusion

The experimental results obtained by our new machine learning system compare very favorably with the results obtained by using a regular pattern language in the nodes of a decision tree. Both systems deliver results in the lower half of the nineties. These are extremely good prediction results for any classification or identification problem.

Transmembrane identification is just one of the numerous biological and other contexts to which sequence based inductive logic programming can be applied. Inductive logic programming in general, and even more so our specific implementation where background knowledge is in the form of callable C functions as predicates, is a very powerful tool that can be extensively applied to the many sequence related biological classification problems. Many different operators, other than 'contains' and 'abstains' can be studied, as well as different translation mechanisms that are biologically relevant.

This work was carried out with the support of a research grant from ISIS, Fujitsu Laboratories.

We acknowledge with thanks Prof Miyano's generous help in sending us the positive and negative examples that they have used in their experimental studies.

References

Arikawa S, Kuhara S, Miyano S, Mukouchi Y, Shinohara A, Shinohara T 1992 A machine discovery from amino acid sequences by decision trees over regular patterns. *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp 618–625

King R, Muggleton S, Lewis R, Sternberg M 1992 Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Natl. Acad. Sci. USA* 89: 11322-11326

Michalski R S 1983 A theory and methodology of inductive learning. *Artif. Intell.* 20: 111–161

Michalski R S, Carbonell J G, Mitchell T M (eds) 1983 An overview of machine learning. In *Machine learning – An artificial intelligence approach* (Palo Alto, CA: Tioga) vol.1, pp 3–24

Muggleton S H 1991 Inductive logic programming. *New Generation Comput.* 8: 295–318

Muggleton S H, Feng C 1990 Efficient induction of logic programs. *Proceedings of the First International Workshop on Algorithmic Learning Theory* (Tokyo: Ohmsha) pp 368–81

Muggleton S H, King R D, Sternberg M J E 1992 Protein secondary structure prediction using logic-based machine learning. *Protein Eng.* 5: 647–657

Quinlan J R 1986 Induction of decision trees. *Mach. Learning* 1: 81–106

Quinlan J R 1990 Learning logical definitions from relations. *Mach. Learning* 5: 239–266

Quinlan J R 1991 Knowledge acquisition from structured data. *IEEE Expert* 6(6): 32–37

Sakakibara Y, Siromoney R 1992 A noise model on learning sets of strings. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (ACM Press) pp 295–302

Shimozono S, Shinohara A, Shinohara T, Miyano S, Kuhara S, Arikawa S 1993 Finding alphabet indexing for decision trees over regular patterns: An approach to bioinformatical knowledge acquisition. *Proceedings of the 26th Hawaii International Conference on System Sciences*, pp 763–772

Siromoney A, Siromoney R 1993 Local exceptions in inductive logic programming. Presented at the International Workshop on Machine Intelligence, ARL Labs, Hitachi, Japan

Siromoney A, Siromoney R 1995 Variations and local exceptions in inductive logic programming. *Machine intelligence* (eds) K Furukawa, D Michie, S Muggleton (Oxford: Clarendon) vol. 14, pp 211–232