

Genetic Programming for Protein Related Text Classification

Marc Segond
European Center for Soft Computing
Calle Gonzalo Gutiérrez Quirós S/N
33600 Mieres, Asturias, Spain.
marc.segond@sofcomputing.es

Cyril Fonlupt, Denis Robilliard
Laboratoire d'Informatique du Littoral,
Maison de la Recherche Blaise Pascal,
50 rue Ferdinand Buisson - BP 719,
62228 CALAIS Cedex, France.
[fonlupt,robillia]@lil.univ-littoral.fr

ABSTRACT

Since the genomics revolution, bioinformatics has never been so popular. Many researchers have investigated with great success the use of evolutionary computation in bioinformatics [19] for example in the field of protein folding or determining genome sequences. In this paper, instead of using evolutionary computation as a way to provide new and innovative solutions to complex bioinformatics problems, we use genetic programming as a tool to evolve programs that are able to automatically classify research papers as dealing or not with a given protein. In a second part, we show that the attributes that are selected by the genetic programming evolved programs can be used efficiently for proteins classification.

Track: Genetic Programming

Categories and Subject Descriptors

D.1.2 [Automatic Programming]

General Terms

Algorithms

Keywords

Genetic Programming, Bio-informatics, Classification, Proteins, Text-mining

1. INTRODUCTION

Information acquired by biological experiments have helped to expand understanding about cellular biology, specifically about biological functions of proteins. However, exponential increase in the number of proteins being identified and sequenced using high throughput experimental approaches (e.g. gene expression microarrays) leads to a growth in the number of uncharacterised proteins.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Determining protein functions is a central goal in bioinformatics, and it is crucial to improve biological knowledge, diagnosis and treatment of diseases. While biological experiments are the ultimate methods to determine the function of proteins, it is not possible to perform a functional assay for every uncharacterised protein due to time and financial constraints, together with the complex nature of these experiments. Hence, a need for using computational methods to assist the annotation of large amounts of protein data appeared. This is a big opportunity to apply data mining techniques to analyse and extract knowledge from biological databases.

Evolutionary algorithms use the principle of natural evolution (as described by Darwin) to evolve solutions to complex problems the same way nature solves problems using, for example, species mutations to adapt to a variation of their environment. In these methods, potential solutions to a problem are evolved to make optimal ones emerge. Evolutionary algorithms have already been successfully applied to biological problems, for example to discover protein functions or interactions, or to classify proteins (see [3, 4]), as well as text mining problems (see [5]) where such algorithms have been used in text mining engines for several applications.

Genetic Programming (GP) is a particular case of Evolutionary Algorithms where the evolved individuals are programs. Evolving rules using GP have been also successfully applied to text classification as can be seen in [6, 7, 8]. The problem we tackle here can be considered as being part of both text mining and protein study as it consists in classification of protein related documents.

First, we introduce in more details the bioinformatics problem. In a second step, we talk about some preprocessing techniques we use to get a suitable dataset for our algorithm. We briefly recall the main principles of Genetic Programming in section 4. Then in section 5, we talk about ROC curves that are used as an optimisation criterion for our learning task. In section 7, we discuss how attributes have been synthesised using the Genetic Programming results. Finally, we draw some conclusions in section 8.

2. DESCRIPTION OF THE PROBLEM

We work here with a set of abstracts extracted from the UniProt database (see [2]). UniProt (Universal Protein Resource) is the world standard non-redundant database of protein annotation. The UniProt database consists of three main database layers: UniProt Archive (UniParc), UniProt

Knowledge base (UniProtKB) and UniProt Reference Clusters (UniRef). The UniProtKB layer consists of two sections: Swiss-Prot and TrEMBL. Swiss-Prot is the richest annotated protein sequence database, containing manually annotated/curated records, with extensive database cross-references and literature citations. TrEMBL database contains computationally analysed records that await full manual annotation.

The dataset we extract from this database contains article abstracts classified as related to proteins and so called *GO terms*. GO terms come from the Gene Ontology databases. They are terms that group proteins together accordingly to their similar functions (e.g. the GO term GO:0005216 which represents the “ion channel activity”). GO terms are organised hierarchically, and one node can have more than one parent. Figure 1 shows the hierarchy of GO terms we work with. Documents belonging to a particular class automatically belong to upper classes (e.g.: a document in class 0005216 will belong also to classes 0015268, 0015267, 0015075, 0005215, 0003674 and 0003673).

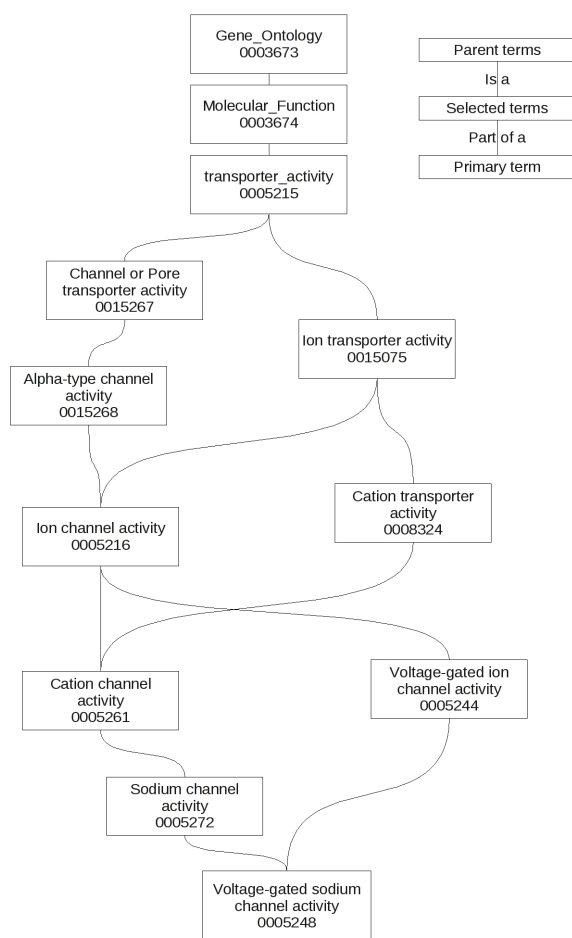


Figure 1: Organisation of the GO terms.

Our dataset is structured in three layers. The first layer is composed of folders whose names are the names of the GO terms. The second layer is also composed of folders whose names are the names of proteins contained in the GO

terms. The third layer is composed of files, each containing an abstract of an article related to a specific protein. This structure is illustrated by Figure 2.

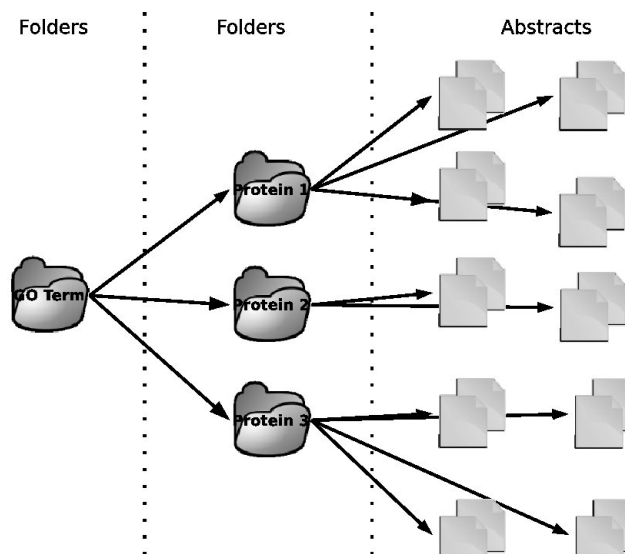


Figure 2: Structure of the dataset.

Using this dataset, the purpose here is to build binary classifiers that are able to determine if a document talks about a specific GO term or not. This means that, as a first work, we need to build as many classifiers as we have GO terms. Genetic Programming is used to automatically evolve these classifiers.

3. PREPROCESSING

Before training the GP classifiers, some preprocessing is necessary on the dataset. We first apply two classical methods in Text Mining that are *Tokenizing* and *Stemming*. Tokenizing is the process of demarcating and possibly classifying sections of a string of input characters. It is performed using a dedicated Java class and stop words removing is processed using the Bow¹ ([1]) library’s list. Stemming is the process for reducing inflected (or sometimes derived) words to their “stem”, base or root form. The stem needs not be identical to the morphological root of the word. The algorithm has been a long-standing problem in computer science and the first paper on the subject was published in 1968. The process of stemming is useful in search engines for query expansion or indexing and other natural language processing problems. Our stemming step is performed using Porter’s algorithm ([17]). As an example of stemming, we can consider the words “stems”, “stemmed” and “stemming” in an abstract. These three words would be reduced to their root form, which is the word “stem”. Now that our documents are tokenized and stemmed, we build a corpus containing all the stems encountered in the documents of our dataset.

The second step is to build a vector for each document. This vector’s length will be equal to the number of stems in the corpus (containing all the stems found in the dataset),

¹<http://www.cs.cmu.edu/~mccallum/bow/>

and each entry will be the occurrence frequency of the stem in the document as given by Equation 1.

$$freq_{stem} = \frac{\text{number of occurrences of stem}}{\text{number of words in the abstract}} \quad (1)$$

We also need to memorise the hierarchy of the GO terms in order to build our sets of vectors and decide if they are positive or negative cases. For example, if a vector (*i.e.* a document) belongs to the GO term GO:0005261, it is also a positive case for all the GO terms that are above in the hierarchy (for example, for the GO term GO:0015268). On the other hand, if a vector does not belong to a given GO term, it will not belong neither to all the GO terms below in the hierarchy.

4. THE GP ALGORITHM

Once these preprocessing steps are performed, the dataset is ready to be used to evolve GP individuals. We use here a GP algorithm that produces real valued functions that will be converted into binary classifiers using a threshold (if the return value is above the threshold, the result is assumed to be “true”, otherwise it is assumed to be “false”). The grammar includes addition, subtraction, multiplication, and a Max node that returns the higher value between two reals. We also use 2 types of terminal nodes: real Ephemeral Random Constants (ERCs), and a set of $freq_i$ nodes to access the occurrence frequency of stems i in the documents. Table 1 summaries the terminal and function set.

Table 1: Nodes set of the GP algorithm

Node	input	output	Function
Mul	2 reals	1 real	multiplication of 2 reals
Sub	2 reals	1 real	subtraction of 2 reals
Add	2 reals	1 real	addition of 2 reals
Max	2 reals	1 real	maximum of 2 reals
$Freq_i$	null	1 real	occurrence frequency of stem i
ERC	null	1 real	random ephemeral constant

Our experimental dataset is composed of 1070 abstracts divided into 6 GO terms. A document belonging to a GO term will also belong to all parents GO terms according to the hierarchy shown in Figure 1. This brings us to a number of abstract for each GO term varying from 80 for the bottom GO terms to more than 200 for the top ones.

The fitness function chosen here is the area under the ROC curve which will be described in the next section. For each individual, the AUC (Area Under the ROC Curve) is computed and the higher it gets, the best the individual is (the AUC is normalised between 0 and 1).

From a more technical point of view, our method was implemented using the ECJ² library, written in the Java language.

5. ROC CURVES

ROC curves were introduced in the 1950’s, being used to optimise noisy radio signals. They are now used in many other domains, and particularly for medical diagnosis for

which they are very efficient (see [9, 10, 11]). The area under the ROC curve is another value that allows the evaluation of the efficiency of a classification method. ROC curves (Receiver Operating Characteristic) allow to study the variations of specificity and sensibility of a numerical test for different discrimination threshold values. In medical diagnosis, these curves allow to get a representation of the ratio between False Positives (FP) and True Positives (TP) depending on the way the diagnosis is interpreted (ROC criterion).

To draw a ROC curve, the x-axis must represent the variable “1 – specificity” (false positives rate) and the y-axis must represent the sensibility (true positives rate). The curve plots the true positives rate in function of the false positives rate for each threshold value (see Figure 3).

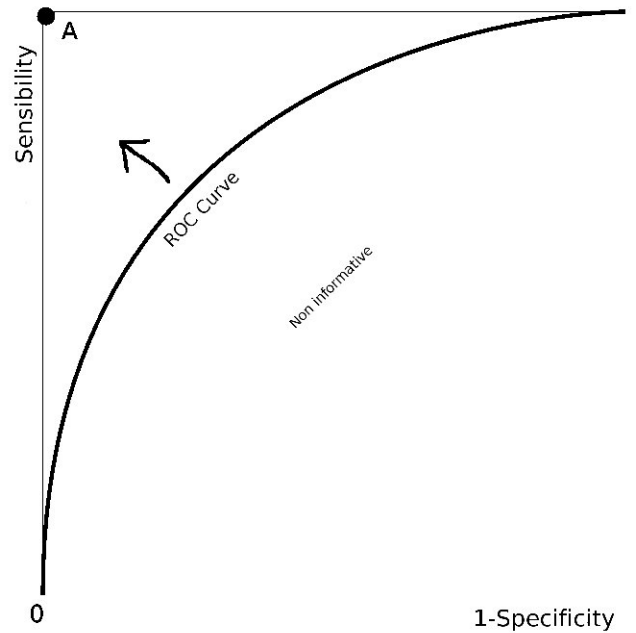


Figure 3: An example of a ROC curve.

Two particular cases of ROC curves can be encountered: the case of an ideal threshold and the case of a non informative test, that is to say random discrimination. A threshold value is ideal if it allows to totally separate all the positive cases from the negative ones, without any false positives or false negatives. It is therefore defined by a sensibility and a specificity equal to 1 (point A of Figure 3). For a non informative test, we are on the diagonal beginning at the point 0.0. If the threshold does not allow to separate positives from negatives, the proportion of positive classified cases is the same in the true positive cases and true negative cases ($sensibility = 1 - specificity$). The corresponding curve is therefore represented by the diagonal of Figure 3. The nearer the ROC curve to the point A, the more informative the test will be. Thus the area under the ROC curve can be considered as a global estimator of a test quality: if the test is non informative, the area is 0.5. If the test is perfectly discriminative, the area will be 1. This area can therefore be used as a learning criterion to be maximised, as exposed

²<http://cs.gmu.edu/~eclab/projects/ecj/>

in [12, 13]. Increasing the area under the ROC curve in optimisation problems allows to increase learning stability in the way that it is able to solve problems caused by the non homogeneity of the learning set ([14]). Technically speaking, the area under the ROC curve is computed easily using SEBAG *et al.* algorithm ([15]), that was corrected in [16].

To be able to use this criterion, we use GP individuals (i.e. functions) that return a real value as output. To get a binary classification, we only need to apply a threshold on this value. This gives the possibility to choose a trade-off between sensibility and specificity, i.e. moving the threshold on the ROC curve.

6. RESULTS OF THE GP

Parameters used for the GP algorithm are shown in Tables 1 and 2.

Table 2: Parameters of the GP algorithm

Parameter	Value
Number of generations	50
Number of individuals per generation	60000
Max tree depth	17
Mutation rate	5%
Crossover rate	90%
Reproduction rate (with elitism = 1)	10%

The curves of Figure 4 show the results of the 10 fold cross-validation on the 6 Go Terms. The 10 fold cross-validation is performed by dividing the set of examples in 10 folds, then learning on nine fold and validating on the one left. By repeating this procedure 10 times, changing at each time the learning set and validation sets, we finally have used the whole example set for learning. As we can see, GP reaches a ROC area of 0.93 in 50 generations of 60000 individuals (2 hours calculations on 16 itanium2 processors). These curves also show the variance of the fitness value averaged over the 10 runs of cross-validation. As we can see, convergence is quite fast, reaching a negligible variance before the 10th generation.

To evaluate the efficiency of our method, we test it against the well known C4.5 decision tree algorithm [18] (available as J48 in the Weka³ toolkit). The confidence parameter used with the C4.5 was left to its default value, that is 0.25. As we can see on the ROC curves of Figure 5, for all the GO terms, GP is equal or dominating the C4.5 algorithm.

These curves show that our GP algorithm is efficient and produces good classifiers. These classifiers output a real value “diagnosis” that allow users to choose the trade-off between sensibility and specificity, via the value of a threshold. However these are less readable than decision trees, as the average size is about 488 nodes per GP tree. Nonetheless we are working on analysing the stems that are present in the best individuals: building statistics about these stems could help to identify words that are significant in identifying a particular protein and might me helpful for biologists.

7. ATTRIBUTES SELECTION

Considering the good results achieved by the GP algorithm, we think that stems used by GP individuals should

³<http://www.cs.waikato.ac.nz/ml/weka/>

be relevant and might be considered as significant attributes for proteins to be classified.

In order to make a comparison, we have considered three different types of protein representations to be used as predictor attributes, namely amino acid composition, protein interaction data and textual information derived from MEDLINE document references.

Amino acid composition attributes were derived directly from proteins’ primary sequences by calculating the ratio between amino acid occurrences and the sequence length. For instance, if the amino acid A (Alanine) occurs 14 times in a protein sequence of length 140, the value of attribute A (attribute representing the composition of amino acid A) would be 0.10 (14/140). In other words, for each amino acid out of 20 that can be found in a protein sequence, we compute the percentage of the sequence composition relative to the amino acid in question. Using this procedure, 20 numeric attributes were produced for each protein in the dataset.

Protein interaction attributes are useful because many proteins interact with one another to perform their function, by assembling multi-protein complexes or metabolic pathways, for example. If one can establish an interaction between a known function protein and an unknown function protein, the protein-protein interaction information can be used to predict the function of the unknown function protein. Protein interaction data was encoded as binary attributes as follows. First, for each protein we retrieved the list of interactor proteins from IntAct database. Then, the complete list of interactors (interactors of all proteins in our dataset) was filtered to remove interactors that only interact with one protein in the dataset. This restriction was necessary to remove interactors present only in one protein, which do not have any predictive power. Finally, the filtered list of interactors (2095 interactors in total) was encoded as a binary attribute vector. Each position of the vector indicates, with a “yes” or “no” value, if the protein interacts with a particular interactor protein or not.

Textual information attributes derived from MEDLINE documents (titles and abstracts) in the form of keywords were encoded as binary attributes, using document references found in proteins’ UniProt records. In total, 1010 documents were linked to the 147 proteins of our dataset. We applied a Genetic Programming (GP) algorithm, as detailed before, to select relevant words (keywords) from linked documents in a preprocessing step. The selected keywords were encoded as a binary attribute vector. Each position of the vector indicates, with a “yes” or “no”, if the documents linked to the protein contain a particular keyword or not. Each of the above three types of attributes was used to produce a single dataset, namely “AA” (for amino acid composition attributes), “PI” (for protein interaction attributes) and “TX” (for textual information attributes).

The idea in this last representation is to build, for each GO Class, a set of keywords used by the best GP individuals. Therefore, for each GO Class, all the keywords used by the five best individuals generated by the GP are extracted, and those that appear at least in two of the five individuals are kept as significant attributes. This allows to reduce the list of keywords from several hundreds to a dozen.

All experiments were conducted running the well-known 10-fold cross-validation procedure. The results concerning

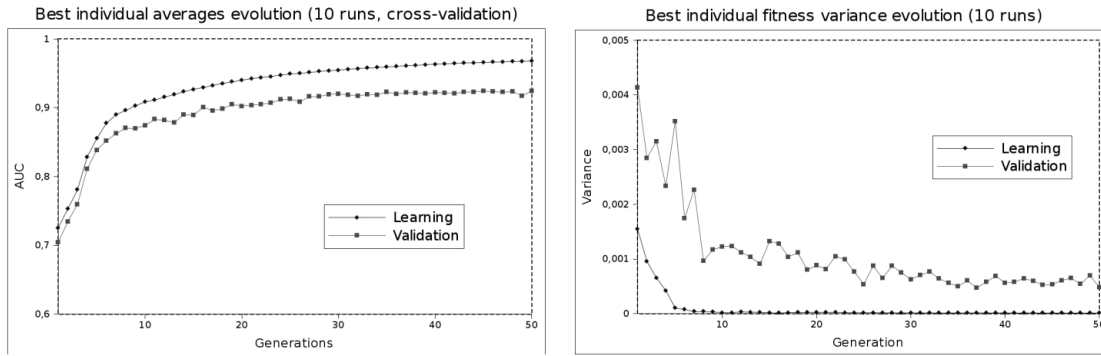


Figure 4: Evolution of the fitness value (Area under the ROC curve) and variance of the best individuals over the 6 GO terms.

predictive accuracy of single attribute datasets per GO term are shown in Table 3. An entry in the column “TX” is shown in bold if, for the corresponding GO term, the accuracy achieved with textual information attributes is significantly greater than the accuracy achieved with the second best type of attribute (columns “AA” or “PI”) for the GO term in question - according to a two-tailed Student’s t-test with significance level $\alpha = 1\%$. Overall, the highest predictive

Table 3: Predictive accuracy (average \pm standard deviation) of single attribute datasets per GO term. An entry in the column “TX” is shown in bold if, for the corresponding GO term, the accuracy achieved with textual information attributes is significantly greater than the accuracy achieved with the second best type of attribute (columns “AA” or “PI”) for the GO term in question - according to a two-tailed Student’s t-test with significance level $\alpha = 1\%$.

Term	AA	PI	TX
GO:0015267	69.26 \pm 2.01	55.30 \pm 3.77	94.38 \pm 1.78
GO:0015075	71.49 \pm 4.37	59.30 \pm 1.14	88.49 \pm 2.81
GO:0005342	84.32 \pm 3.01	85.75 \pm 2.08	87.18 \pm 1.49
GO:0005275	88.57 \pm 2.80	87.93 \pm 1.53	92.65 \pm 1.52
GO:0015268	65.92 \pm 3.29	57.02 \pm 2.64	93.24 \pm 2.98
GO:0008324	64.55 \pm 3.76	59.92 \pm 1.63	83.77 \pm 2.24
GO:0008509	92.01 \pm 1.86	93.21 \pm 1.42	93.44 \pm 1.87
GO:0046943	86.59 \pm 2.38	89.89 \pm 1.78	88.55 \pm 1.35
GO:0005216	73.32 \pm 3.46	57.71 \pm 3.60	93.18 \pm 1.77
GO:0015171	88.59 \pm 1.62	89.26 \pm 1.74	88.51 \pm 2.27
GO:0005261	71.92 \pm 2.17	68.68 \pm 2.66	88.93 \pm 3.44
GO:0015276	86.37 \pm 2.00	89.18 \pm 1.03	85.18 \pm 2.34
GO:0005244	69.42 \pm 1.95	81.05 \pm 3.02	83.58 \pm 2.58
GO:0005253	94.03 \pm 2.01	94.65 \pm 1.85	97.99 \pm 1.02
GO:0005267	82.38 \pm 1.75	88.52 \pm 0.95	97.29 \pm 1.11
GO:0005262	86.90 \pm 2.64	85.64 \pm 1.67	87.42 \pm 3.15
GO:0005245	89.90 \pm 2.84	93.24 \pm 2.22	90.57 \pm 2.84

accuracy was achieved when using textual information as predictor attributes (dataset “TX”) - 7 out of 17 cases with no significant losses in the remaining 10 cases. These results indicate that textual information attributes are useful

for predicting GO terms at any level in the hierarchy used in our experiments. Protein interaction attributes achieved the lowest predictive accuracy in 5 out of 17 cases - in GO terms GO:0015267, GO:0015075, GO:0015268, GO:0008324 and GO:0005216. At the same time, they achieved competitive predictive accuracy (when compared to textual information attributes) in 3 cases - GO terms GO:0008509, GO:0005253 and GO:0005245. These results suggest that protein interaction attributes are useful for predicting GO terms at levels near the leaves of the hierarchy, since at levels near the root of the hierarchy they achieved a significantly lower accuracy. Amino acid composition attributes achieved an average predictive accuracy in overall. In 2 cases - GO terms GO:0008509 and GO:0005253 - the accuracy achieved was competitive with textual information attributes.

8. CONCLUSION

It is well known by researchers in biology that the numbers of papers published is now so large that trying to be up to date is a very difficult goal to achieve. In this paper, we show that evolutionary computing can be used to automatically analyse papers abstracts and to automatically classify them. In a second part, we show that stems obtained by the evolutionary algorithms can be used as attributes for proteins classification, with an efficiency at least equal, and sometimes better, than the efficiency of the classical attributes used in this field.

9. REFERENCES

- [1] McCallum, A. K.: Bow: A toolkit for statistical language modelling, text retrieval, classification and clustering. Arch. Rat. Mech. Anal. **78** (1996) 315–333
- [2] R. Apweiler, A. Bairoch, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O’Donovan, N. Redaschi, and L.L. Yeh. : Uniprot: the universal protein knowledgebase. Nucleic Acid Research, **32**:D115–D119 (2004).
- [3] A. Chan and A.A. Freitas: A new ant colony algorithm for multi-label classification with applications in bioinformatics. Proc. Genetic and Evolutionary Computation Conference (GECCO-2006) (2006) 27–34, ACM Press.

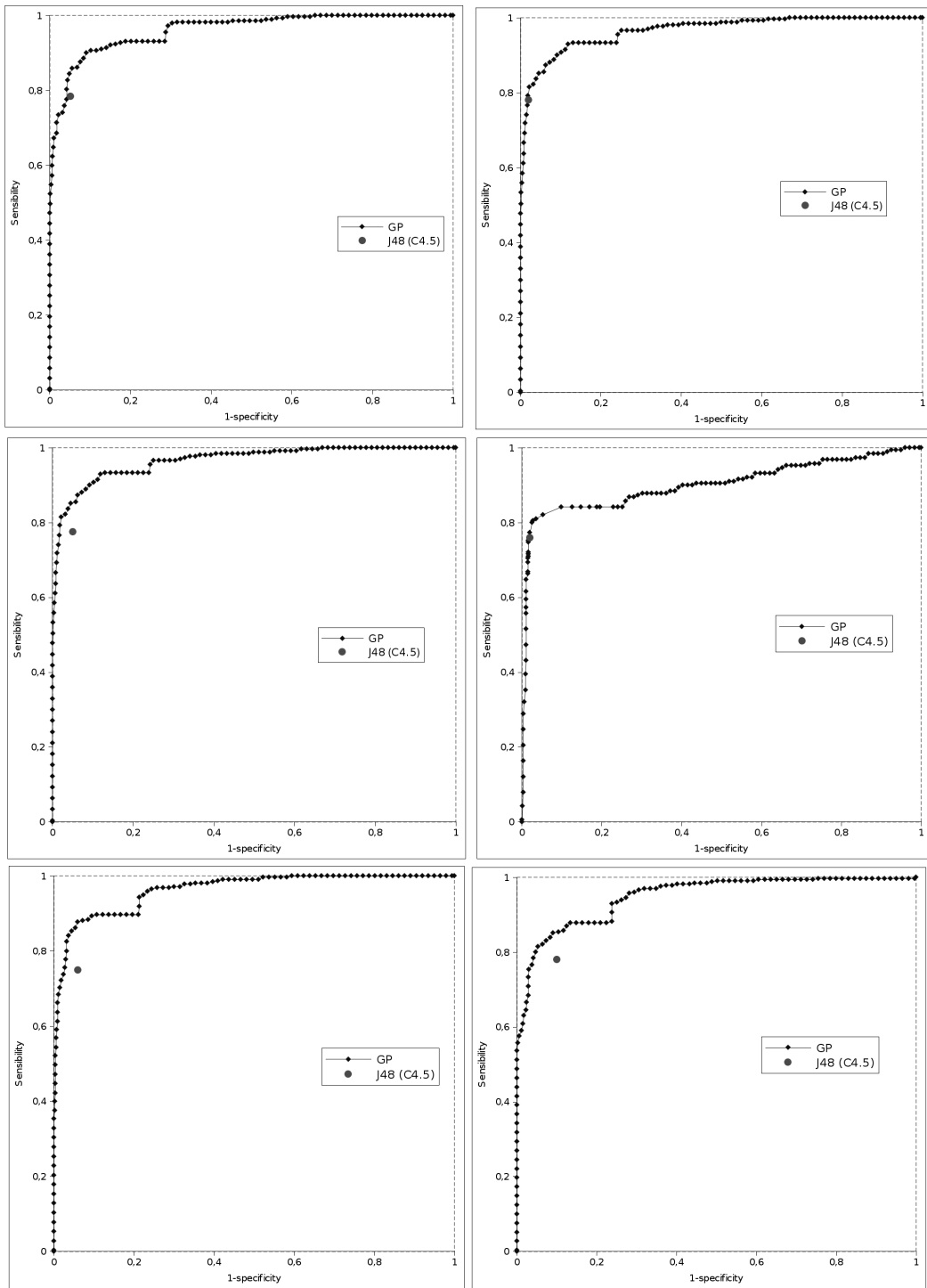


Figure 5: Comparison between the C4.5 algorithm and the GP classifiers for 6 GO terms. Curves shown here are average ROC curves of the best individuals over the 10 runs on the validation set.

- [4] DF Tsunoda and HS Lopes and AA Freitas: An evolutionary approach for motif discovery and transmembrane protein classification. Applications of Evolutionary Computing (Proc. of EvoBIO-2005: 3rd European Workshop on Evolutionary Bioinformatics) (2005) 105–114, Springer.
- [5] C.N. Silla Jr. and G.L. Pappa and A.A. Freitas and C.A.A. Kaestner: Automatic text summarization with genetic algorithm-based attribute selection. Advances in Artificial Intelligence (IBERAMIA 2004, Proc. 9th Ibero-American Conference on AI (2004) 305-314, Springer-Verlag.
- [6] Laurence Hirsch, Masoud Saeedi, Robin Hirsch: Evolving Text Classification Rules with Genetic Programming. Applied Artificial Intelligence **19** (2005) 659–676.
- [7] Laurence Hirsch, Masoud Saeedi, Robin Hirsch: Evolving Rules for Document Classification. Proceedings of the 8th European Conference on Genetic Programming **3447** (2005) 85–95, Springer-Verlag.
- [8] Baoping Zhang: Intelligent Fusion of Evidence from Multiple Sources for Text Classification. PhD thesis, Virginia Polytechnic Institute and State University (2006).
- [9] Hanley, J. and McNeil, B.: The meaning and use of the area under the receiver operating characteristic (roc) curve. Radiology **143** 29–36
- [10] Metz, C.: Basic principles of roc analysis. Semin. Nuclear Med. **VIII** (1978) (4):283–298.
- [11] Langdon, W. and Buxton, B.: Evolving receiver operating characteristics for data fusion. 4th European Conference, EuroGP 2001 (2001) 87–96.
- [12] Ferri, C., Flach, P. A. and Hernandez-Orallo, J.: Learning decision trees using the area under the roc curve. Proceedings of the 19th International Conference on Machine Learning (2002) 179-186.
- [13] Mozer, M., Dodier, R., Colagrosso, M., Guerra-Salcedo, C. and Wolniewicz, R. Prodding the roc curve : Constrained optimization of classifier performance. Advances in Neural Information Processing Systems, (2001), The MIT Press.
- [14] Bousquet, O. and Elisseeff, A.: Stability and generalization. Journal of Machine Learning Research **2** (2002) 499–526.
- [15] Sebag, M., Aze, J. and Lucas, N: Roc-based evolutionary learning : Application to medical data mining. Artificial Evolution (2003) 385–396, Springer.
- [16] Segond, M., Robilliard, D. and Fonlupt, C.: Iterative filter generation using genetic programming. Euro'GP 2006, volume 3905 of LNCS (2006) 145–153, Springer.
- [17] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter: New models in probabilistic information retrieval. London British Library. British Library Research and Development Report **5587** (1980).
- [18] Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993).
- [19] G. Fogel, D. Corne: Evolutionary Computation in Bioinformatics. Morgan-Kauffman, 2002.