Classification of Domains with Boosted Blast

Cécile Capponi[1], Gwennaele Fichant[2], Yves Quentin[2], and François Denis[1]

[1] LIF - CNRS, Université de Provence, 39, avenue Joliot Curie
13453 Marseille Cedex 13, France
(e-mail: capponi@cmi.univ-mrs.fr, fdenis@cmi.univ-mrs.fr)
[2] LMGM - IBCG - CNRS, Université Paul Sabatier, 118, route de Narbonne,
31062 Toulouse Cedex, France
(e-mail: fichant@bibcg.biotoul.fr, quentin@ibcg.biotoul.fr)

Abstract. This paper presents the first real experimentations of the boosting techniques applied to Blast for producing a model of functional domains whose amino-acids primary sequences are not conserved during evolution. The BlastBoost algorithm is depicted, and first results are analysed, showing the relevance of our approach.
Keywords: Bioinformatics, Boosting, Sequence similarity, Functional domains.

1 Introduction

The function of a single protein is mainly carried out by a *domain* which is a subsequence of amino-acids within the whole sequence of the protein. During evolution, the sequence of such a domain can be significantly modified while the function is still conserved. One way of predicting the function of a protein is to identify a known domain in the protein, despite sequence modifications such as deletions or substitutions. Domains can be grouped in functional families, themselves subdivided in subfamilies. Our work deals with functional families whose domains are not well conserved during evolution, which means that, given the sequence of a protein, it is hard to predict whether it carries the domain associated with the function. Formally, let F be a functional family, let $P = \{p_1, ..., p_n\}$ a set of annotated proteins which are known to belong or not to F, our problem is to decide whether any new protein p belongs to F. It is a supervised binary classification problem: how to build a rule from the annotated proteins of P in order to determine the class of new unannotated proteins.

In many cases, comparing a new sequence of protein p with some sequences of the family F is enough for predicting whether $p \in F$. Such a similarity search may be achieved by using either an alignment program such as Blast [Altschul *et al.*, 1997] or any model of the family's sequences, for example stochastic and probabilistic models such as Hidden Markov Models. Unfortunately, none of these methods is satisfactory whenever the sequences of the domains of the family are not conserved: the models are hard to build. For example, Membrane Spanning Domains (MSD) play the role of a pore through which a substrate goes in and/or out of the cell. The composition in

amino-acids of such a domain depends essentially on the nature of membrane of the considered species, so their sequences are not conserved. As a consequence, using usual alignment programs, or any current probabilistic model, is not satisfactory for retrieving such a domain onto a protein.

One way for identifying such unconserved domains of a family F on a new protein p is to successively (1) specialize the family F into several subfamilies, (2) search significant similarities between p and each known protein of each subfamily, and (3) check back the obtained predictions in order to remove false positives. It is the IRIS strategy, as presented in [Quentin *et al.*, 2002] for retrieving proteins that carry a MSD domain. Despite good results, such a strategy is tedious to set up for many reasons. Among others, the subdivision of the whole family into many subfamilies is possible only when the considered domain has been widely experimentally studied. Moreover, since it is still hand-made, the subdivision may suffer from annotation mistakes.

Our proposal is to use a classification technique which avoids the subdivision of the functional family: how to learn a rule for deciding whether a given protein's sequence contains a domain of a given functional family. Usual techniques in supervised classification consist in computing at once one *strong* classification rule, *i.e.* a rule that is both selective (few false positives) and sensitive (few false negatives). Instead our proposal is to progressively learn a sequence of *weak* rules: each weak rule is not efficient on the whole training set albeit better than a random prediction. Each weak rule is learnt from an example that was badly classified using the previous weak rules. A strong rule is eventually computed by combining weak rules weighted by the confidence we have on each. Such a technique is named *boosting* [Schapire, 1990] [Freund, 1995]. Our proposal is to learn those classifiers using BLAST, which is an algorithm to produce and evaluate local sequence alignments based on a stochastic model.

2 Boosting blast

In the following of the paper, let X be the instance space and $Y = \{-1, +1\}$ be the label set. A learning algorithm takes as input a training set $S = \{(x_1, y_1) \cdots (x_n, y_n)\}$ where $x_i \in X$ and $y_i \in Y$. The learnt classifier is a function $H : X \mapsto Y$ that predicts the label of any example of X according to a model computed from the training set S. The training error of H is the error rate made on the training set, while the test error of H is an approximation of the real error rate made by H on all the instances of X. The IRIS strategy starts from observations which are positive examples described by their sequence of amino-acids, and applies a combination of alignment programs in order to tag any new protein. Our approach is to consider the problem as a classification problem which must be solved by taking advantage of the predictive power of local alignment programs.

Since boosting is a general method for improving the accuracy of any given learning algorithm, we propose to use it in order to improve the significancy of learning algorithms computed over local sequences alignments. We chose to boost BLAST as it is the most popular tool for investigating sequence alignments, and because it relies on a stochastic model.

2.1 Boosting: principles and algorithm

Boosting is a machine-learning method which is based on the observation that finding many moderately inaccurate rules of thumb can be a lot easier than finding a single, highly accurate prediction rule. Let M be an algorithm or a method for finding the rules of thumb: let name it a "weak" or "base" learning algorithm. The boosting approach calls M repeatedly, each time feeding it with a different subset of S, more precisely a different distribution over S. Each time M is called, it generates a new weak prediction rule; after many rounds, the boosting algorithm combines these weak rules into a single prediction rule that is proven to be much more accurate than any one of the weak rules when enough data is available [Schapire, 1990]. On each round, the distribution of S is updated in such a way that the weight on the examples misclassified by the preceding weak rule is increased: this forces the base learner M to focus its attention on the "hardest" examples. The final combination of the weak rules is a simple weighted majority vote of their predictions: the weight asssigned to a weak rule should actually account for the confidence one can have on it.

We focus here on the AdaBoost algorithm (*cf.* Algorithm 1 further on, introduced by [Freund and Schapire, 1997]), which is of reference. A complete and easy presentation of practical and theoretical results about boosting and AdaBoost is available in [Schapire, 2002], especially results concerning error bounds.

Algorithm 1 AdaBoost(T), where T is the number of rounds (iterations)

Given: $(x_1, y_1), \cdots, (x_n, y_n)$ where $x_i \in X$ and $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) \leftarrow 1/n, \forall i \in 1..n$ (D is indexed by the indices of the examples)
for all $t = 1, \cdots, T$ **do**
  Train base learner M using distribution D_t
  Get the base classifier $h_t : X \rightarrow \{-1, +1\}$
  Compute $\alpha_t \in \mathbb{R}$
  Update:

$$D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

  where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution of probabilities.
end for
Output the final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Let us here comment the Algorithm 1 which considers the simplest case: the range of each h_t is binary. D is a distribution over the sample S which is updated at each iteration t. Let $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$ be the training error of the base classifier h_t. The parameter α_t should measure the importance assigned to h_t: it is usually related to ϵ_t. For binary base classifiers, we typically set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$ as suggested in [Schapire, 2002].

2.2 Aligning sequences with Blast

For finding similarities between protein sequences, some algorithms compare, in a pairwise fashion, a *query* sequence to all the sequences of a specified database. Each comparison is given a *score* reflecting the degree of similarity between the sequences. The similarity is measured and shown by *aligning* two sequences, globally or locally. A global alignment is an optimal alignment that includes all characters from each sequence, whereas a local alignment is an optimal alignment that includes only the most similar local region(s) (e.g. [Smith and Waterman, 1981]).

Among these algorithms, heuristic algorithms such as Blast and Fasta trade reduced accuracy for improved efficiency. Blast [Altschul *et al.*, 1990] is actually a set of sequence comparison algorithms that are used to search sequence databases for optimal local alignments to a query. Blast improves the overall speed of searches while retaining good sensitivity by breaking the query and database sequences into fragments (words), and initially seeking exact matches between fragments. The algorithm then tries to significantly raise the length of each match: the obtained extended fragments are named high-scoring segment pairs (HSPs). Hence, each pairwise sequence alignment is first assigned a raw score S (which accounts for the score of its HSP). Then, if the raw score is over a given threshold, a statistical score is computed in such a way one can discriminate between real and artefactual matches: the expected number of HSPs with score at least S is given by : $E = Kmn \exp^{-\lambda S}$ where K and λ are parameters of the scoring system (gap costs and the matrix of amino-acids substitutions), and m and n are the lengths of the sequences. This score E, named the e-value of an alignment, is the expected number of chance alignments with a score larger than (or equal to) S [Altschul *et al.*, 1997]. The smaller the e-value is, the most significant the alignment is.

Let $\mathcal{A}(x, D, \tau)$ be a formatted result of the `blastp` program (program of the Blast software for aligning proteins sequences) with x as a query, and D as the database: it is a set that contains all the proteins of D aligned with x with an e-value less than τ.

2.3 Boosting Blast

Experimentations of the Iris strategy show that some functional subfamilies of MSD are more difficult than others to be delimited. Consequently, our

major intuition was that the boosting principle should help to slim over the covering of the description space of the subfamilies, by focusing on the proteins that are on the boundaries hence improving the whole covering of each subfamily. Indeed, many *false positives* are produced by all the tested recognition methods (HMM, profiles, etc.) while they could help to make more accurate the model of the subfamilies We then supposed that the boosting techniques would help to focus on proteins of the boundary of each subfamily.

The Algorithm 2 depicts the backbone of "boosting blast" for computing a model of one type of functional domains. At each iteration of the boosting algorithm, the obtained weak classifier h_t is a decision tree, built from a BLAST alignment of sequences whose query has been randomly selected in the learning sample with respect to the distribution D_t. In order to compute a decision tree of deepness 1 (a stump), a BLAST program is launched with query x_t over the set of known protein's sequences X, which leads to the set of proteins aligned with x_t under the given threshold τ of e-value: $\mathcal{A}(x_t, X, \tau)$. The base classifier h_t generated at iteration t is then obvious: for any sequence of protein $x \in X$, if $x \in \mathcal{A}(x_t, X, \tau)$ (*i.e.* x is aligned with x_t) then x is tagged as x_t, otherwise it is classifed like the majority (according to D_t) of the learning examples which are out of $\mathcal{A}(x_t, X, \tau)$. With such a decision tree, we can expect that the training error of each weak classifier is usually less than 0.5 (which is a condition of the boosting technique). Indeed, the BLAST model of alignments is usually very predictive locally, so a few errors should be observed for examples aligned with x_t; moreover, since the majority class is chosen for classifying proteins that are not aligned with the x_t, the training error over them is less than 0.5. We expect that boosting would then extend the local predictivity of BLAST to a global predictivity.

We actually set up many different kinds of weak classifiers. Two possible variants among others are:

1. *the deepness of the decision trees.* The algorithm 2 considers trees with only one test. The generalization to decision trees with d tests (a comb) is straightforward: if the test j does not lead to a valuable alignment, another example is selected from the same distribution D_t from which a new test $j + 1$ is achieved, and so on until d `blastp` have been launched with d different queries of the sample. Such a generalization should raise the number of examples covered by each weak classifier, hence hopefully decrease the training error ϵ_t.
2. *The class of the selected example.* If considering all proteins of a species, the ratio between positive and negative proteins is very low. As a consequence, we chose only negative examples which are known to be close to the positive examples. Then, since both classes may be randomly selected, we authorize to allow different thresholds τ_+ and τ_- for the e-value, whether the query is a positive or a negative example. An important variant is to only authorize the selection of positive examples: in

Algorithm 2 BlastBoost(τ,T)

Given: $(x_1, y_1), \cdots, (x_n, y_n)$ where $x_i \in X$ and $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) \leftarrow 1/n$
for all $t = 1, \cdots, T$ **do**
Select $x_{i,t}$ according to the distribution D_t
Compute $\mathcal{A}_t = \mathcal{A}(x_{i,t}, X, \tau)$ with `blastp`
Get $h_t : X \rightarrow \{-1, +1\}$ such that $\forall x \in X$:

$$\text{if } x \in \mathcal{A}_t \text{ then } h_t(x) = y_{i,t} \text{ else } h_t(x) = \text{argmax}_{k \in \{-1,+1\}} \sum_{j, y_j = k, x_j \notin \mathcal{A}_t} D_t(j)$$

Compute

$$\epsilon_t = \sum_{i=1..n, h_t(x_i) \neq y_i} D_t(i) \text{ and } \alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Update:

$$D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

end for
Output the final classifier:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

such a way, the learning algorithm does not try to learn negative examples, instead it focuses on the boundaries of the family.

3 Experimentation

3.1 Protocols

The performance of the approach has been evaluated on a specific domain found as a component of ABC transporters: the Membran Spanning Domain (MSD). These domains have been chosen because they are poorly conserved in sequence and their identification led to a large number of false positives in previous analysis [Quentin *et al.*, 2002]. We used five genomes to learn the model, and five other genomes to test it. Genomes, both in the learning and the test sets, were chosen according to the phylogeny. These sets are made up of all the positive proteins (i.e. those proteins that carry a MSD domain), and all the negative proteins that are close to the positive ones (i.e. they are known to be aligned with one or more proteins of a MSD subfamily). Each experiment has been launched 10 times with the same parametrization: the reported results of one experiment are actually a mean of the results. These first experiments helped us to investigate the role of parameters (d, τ, etc.).

Ideally, the BlastBoost algorithm should run each BLAST against the whole set of known protein sequences. As it would be too long during both the learning and the testing steps, we precomputed with BLASTP, and stored, the alignement of each sequence of a species with each sequence of all the species.

As far as we known, no other algorithm seeking similarities between proteins have been integrated within a boosting algorithm. As a consequence, we compare the results of BlastBoost with the IRIS strategy which previously subdivided the MSD functional family into 18 subfamilies in order to be able to annotate any new example with a low error test.

3.2 Results and discussion

The figure 1 presents the best results that we obtained by tuning up the set of parameters and alternative algorithms presented section 2.3. The four categories of test share the same $d = 3$; in all categories, positive and negative examples were selected during the learning step.

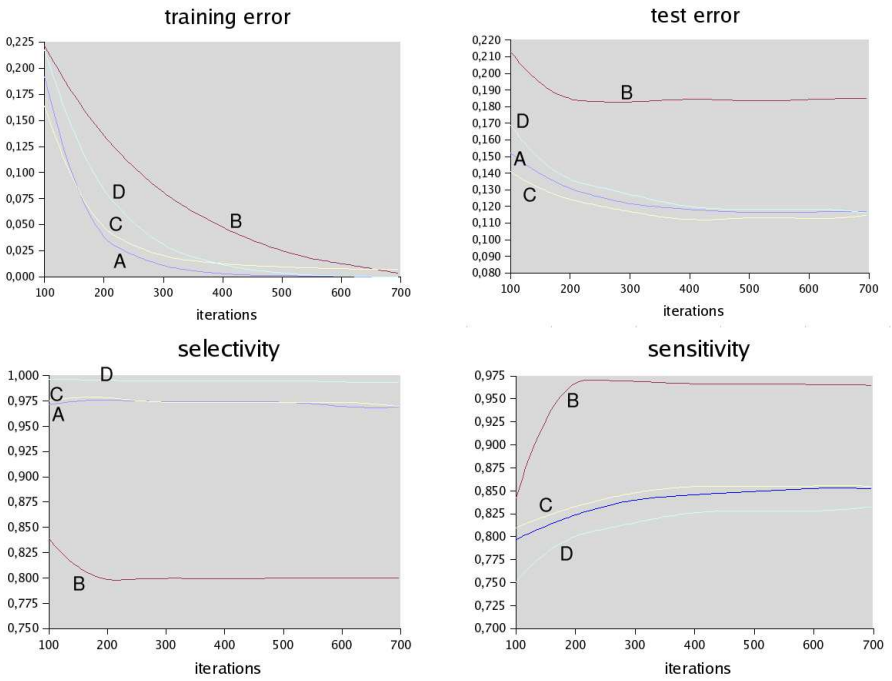

Fig. 1. The number of iterations corresponds to the parameter T in the algorithms. In tests of category A, $\tau_+ = 10^{-2}$ and $\tau_- = 10^{-10}$. In tests of category B, $\tau_+ = 10^{-10}$ and $\tau_- = 10^{-2}$. In tests of category C, $\tau_+ = 10^{-2}$ and $\tau_- = 10^{-2}$. In tests of category D, $\tau_+ = 10^{-5}$ and $\tau_- = 10^{-5}$.

The training (resp. testing) set contains 161 (resp. 172) positive proteins and 73 (resp. 84) negative proteins. The selectivity of our method is almost as good as this of the Iris method applied to MSD, while our sensitivity is better (up to 0.999 with BlastBoost, and 0.946 with Iris). Yet, their method previously subdivides the functional families into 18 subfamilies, so their learning and testing steps are independant from one subfamily to another, therefore more accurate. As a consequence, the results of BlastBoost are good: with comparable results, BlastBoost is efficient on the whole functional family even if the proteins sequences are not conserved. When analyzing the misclassified data, we noticed that the problematic subfamilies identified by Iris (M7 and M9) were better characterized by BlastBoost, while one subfamily has not been circled by BlastBoost whereas it was an "easy" family for Iris (M12). This last point results from an under-representation of M12 members in the learning set (1 out of 161 proteins), leading to false negatives during the tests. So, in order for our result to be statistically significant, we still have to work out the samples so that each subfamily is represented.

We improved our results by increasing the number of genomes in both samples: with eight in each, the test error is less than 0.1 in the categories of test B and C while the selectivity gets perfect in these categories. The accurate study of the produced weak classifiers shows that some sets of aligned sequences have a poor significance in their globality, for example when proteins are dimers. Thus, we think of defining and importing the significancy of an alignment within the boosting model, in addition to the significancy of pairwise alignments (e-value). The InfoBoost algorithm [Aslam, 2000] should be a first step towards the integration of sequences alignements properties within a boosting algorithm, for it pays attention on the quantitative *and* qualitative performance of each weak classifier, which in our case can be measured from the properties of each sequences alignement.

4 Conclusion

We presented a new way for learning a model of unconserved functional families, without dividing them into subfamilies, which is based on amino-acids sequences, by applying the boosting techniques to one performant alignment program, Blast. Our first results are good and promising. We think that several improvments could be carried out, independently from the tuning of the involved parameters. Among other, our first perspective is to integrate in the boosting model, and especially in the rated confidence of weak classifiers, some properties of alignment algorithms such as the density of an alignment, which involves its covering rate of the database and the inner significancy of the e-value's distribution.

References

[Altschul *et al.*, 1990]S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[Altschul *et al.*, 1997]S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psiblast: a new generation of protein database search programs. *Nucleic Acid Research*, 25:3389–3402, 1997.

[Aslam, 2000]J. A. Aslam. Improving algorithms for boosting. In *13th Conference On Learning Theory, COLT'2000*, pages 200–207, Stanford, CA, USA, 2000. Morgan Kaufmann.

[Freund and Schapire, 1997]Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, aug 1997.

[Freund, 1995]Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[Quentin *et al.*, 2002]Y. Quentin, J. Chabalier, and G. Fichant. Strategies for the identification, the assembly and the classification of integrated biological systems in completely sequenced genomes. *Computers and Chemistry*, 26:447–457, 2002.

[Schapire, 1990]R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Schapire, 2002]R.E. Schapire. The boosting approach to machine learning: an overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[Smith and Waterman, 1981]T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.