Learning to Recognize Brain Specific Proteins Based on Low-level Features from On-line Prediction Servers

Mikael Huss Henrik Boström Lars Asker Joakim Cöster

Virtual Genetics Laboratory

Fogdevreten 2a

SE-171 77 Solna, Sweden

{mikael.huss, henrik.bostrom, lars.asker, joakim.coster}@vglab.com

ABSTRACT

During the last decade, the area of bioinformatics has produced an overwhelming amount of data, with the recently published draft of the human genome being the most prominent example. This has enabled researchers to use data driven, rather than hypothesis driven, methods to address a wide variety of specific problems related to the analysis of biological sequences (e.g., protein, DNA and RNA sequences). Today a number of low-level properties of biological sequences, like the presence or absence of signal peptides, can be obtained from publicly available on-line prediction servers. Such a server typically implements a classifier which is trained to determine a single property of a sequence on the basis of various kinds of biochemical laboratory results. In this paper we investigate how the low-level data from these distributed on-line sources can be combined to construct a classifier that recognizes a high-level property, namely the brain specificity, of a protein. This is a task for which no satisfactory method has yet been reported. Features gathered from eight different on-line prediction servers are used in experiments to yield a representation of protein sequences obtained from the Swiss-Prot database. The experiments show that using these low-level features, classifiers can be trained to predict the brain specificity with a surprisingly high accuracy (around 70%).

Keywords

Brain specificity, classification, machine learning

1. INTRODUCTION

Recent years' dramatic increase in the amount of data available on the Internet and in public databases combined with the increasing memory and processing speed of workstations has created new opportunities for researchers to make scientific discoveries. The wealth of biological data that has thus become accessible to the biological research community has irrevocably changed biologists' way of doing science. A new, data-driven paradigm, sometimes labeled "in silico biology", has gained widespread acceptance. Many biologists are now comfortable with the idea of building computer models to support or altogether replace laboratory experiments. In this and many similar fields where experiments have traditionally been performed in vivo or in vitro, new possibilities have arisen to perform experiments in silico. This has led to new trends where researchers in many areas today use computers to run simulations in order to gain insight into the underlying functions and causality of complex systems. The Internet has also created opportunities for more efficient ways of collaborating, doing discoveries, and sharing information. Recently, systems have been developed which allow discoveries to be made involving information distributed over several geographically distributed systems.

During the last decade, a number of computational models have been built for predicting various properties of biological sequences (mainly protein, RNA and DNA sequences). These models often specialize in predicting low-level biological characteristics of sequences and are typically built on training data obtained as a result of laboratory experiments and previously published results. Many of the models are publicly available on the World Wide Web.

One of the recent highlights in the world of science was the publication of the draft human genome sequence [20], [31]. The human genome sequence is a goldmine for those wishing to take full advantage of the previously developed prediction models. At least 10.000 genes are still unknown, and the number of unknown gene products (typically proteins) is even higher [31]. Tools for finding genes and accurately predicting their functions are in high demand.

In this paper we present an approach to harnessing the combined power of some of the public prediction servers by using their predictions of low-level protein features to generate a classifier for recognizing a high-level feature of a protein, namely brain-specificity. Tissue specificity prediction in general is still an unsolved problem. The importance of being able to predict tissue specificity from sequence lays in the fact that tissue specific proteins tend to be good candidates for drug targets. We demonstrate that it is possible to build a classifier that predicts the brain specificity of a protein based on such low level protein features from a selection of such distributed prediction servers. Preliminary experiments show that the combined classifier is able to predict brain specificity with an accuracy of around 70%.

2. DATA SELECTION

The World Wide Web contains a wealth of resources for manipulating biological data. Some examples of such resources are sequence databases where new entries are continuously submitted and annotated by researchers, image databases containing 3D structures of molecules, databases containing known biochemical interactions between different compounds, and prediction servers for inferring characteristics about sequences. These resources are still fairly poorly integrated, although some attempts have been made to construct interfaces that co-submit queries to several of these servers and combine the results into reports.

We chose to retrieve our training data from the public protein database Swiss-Prot. Swiss-Prot is a manually curated database of protein sequences that includes information on protein function, expression, and so on [2]. The dataset consisted of 214 protein sequences. 107 of the sequences were positive (brain-specific) and 107 were negative.

Swiss-Prot entries (sometimes) contain a data field labeled TISSUE SPECIFICITY. All human proteins in Swiss-Prot were scanned (using a Perl script) for the presence of keywords relating to the brain, the nervous system, etc. in this data field. The sequences that were found in this way were then manually inspected to make sure that they really were positive examples.

Negative examples were taken from Swiss-Prot using random selection on human proteins followed by manual inspection of the TISSUE SPECIFICITY data field of each entry. Proteins for which the TISSUE SPECIFICITY field described specificity for tissues or organs other than the brain, or for which it explicitly stated that the protein was not found in the brain, were chosen as negative examples.

A Perl script was developed for contacting all relevant servers, posting the protein sequences, parsing the output from the servers and writing the results to a file.

The choice of input features was constrained by the availability of public prediction servers. The goal was to use all high-quality information about a sequence that can be gathered from the World Wide Web. The features used were:

(1) *Number of amino acids*. This was the only feature that was known to have a possible correlation with brain specificity. Brain-specific mRNAs are one average twice as large as other mRNAs [26], which makes it plausible that the translated brain-specific polypeptides would be longer than other polypeptides.

(2) *Aliphatic index*. This value was obtained from ExPaSy's ProtParam prediction server [22]. The aliphatic index is defined as the relative volume of a protein occupied by aliphatic side chains (alanine, valine, isoleucine, and leucine). The aliphatic index may be regarded as a positive factor for the increase of thermostability of globular proteins [18], but it is not known if it is correlated with protein tissue specificity in any way.

(3) *GalNAc O-linked glycosylation sites*. GalNAc O-linked glycosylation is known to play a role in the immune system, for instance in connection with MHC class I molecules [28]. The number of potential GalNAc sites was obtained from the NetOGlyc program [16] at the CBS prediction server page, created at the Center for Biological Sequence Analysis at the Technical College of Denmark. This number was then divided by sequence length (number of amino acids) in order to get a relative number of potential GalNAc sites.

(4) *Grand average of hydropathicity (GRAVY)*. Will be called simply "hydropathy index" below. Values were obtained from ProtParam, see (2). The common hydropathy index, defined at one specific position in a sequence, is the mean value of the hydrophobicity (tendency to avoid water) of the amino acids within a window, usually 19 residues long, around each position. In transmembrane helices, which are present in many neural proteins, the hydropathy index is high for a number of consecutive positions in the sequence. The "grand average of hydropathicity" is the average value of the hydropathy index at each position. See [19] for more details.

(5) *Instability index*. Certain dipeptides occur more frequently in unstable proteins than in stable ones. The inventors of this index have assigned a weight value of instability to each of the 400 different dipeptides [15]. The values obtained for a whole protein tend to fall within the range 0-100. A protein whose instability index is smaller than 40 is predicted as stable, a value above 40 predicts that the protein may be unstable. It is not known whether instability index correlates with protein function.

(6) *Phosphorylation sites*. Phosphorylation is an extremely important post-translational protein modification. It occurs in many biological processes, including cell signaling, which is an important function in the brain. The number of potential phosphorylation sites was obtained from the NetPhos program [3] at the CBS prediction site; see (3). This number was divided by sequence length to give a relative measure.

(7) *GlcNAc O-linked glycosylation sites*. GlcNAc is another, newly found type of O-linked glycosylation, distinct from GalNAc. This type of glycosylation occurs in nucleocytoplasmic proteins and may play a role in signal transduction, although this is not entirely clear [32]. The number of GlcNAc sites was predicted by the YinOYang program at the CBS site; see (3). This number was divided by sequence length to yield a relative measure. For sequences that were predicted to have a signal peptide, the value of GlcNAc was set to zero. Such sequences are unlikely to be intracellular, and hence unlikely to be O-GlcNAcylated.

(8) *Isoelectric point*. The isoelectric point is the pH value where a protein has no net charge. This value was obtained from ProtParam at ExPaSy; see (2). No known correlations exist between isoelectric point (also written as pI) and nervous system function.

(9) *Alpha helix content*. This value is the predicted percentage of alpha helix positions in a protein, i.e. the percentage of amino acids predicted to be located in an alpha helix. See

(10) *Beta sheet content*. Same as (9), but for beta sheets instead of alpha helices. The predictions for alpha helices and beta

sheets were obtained from SSCP, a server hosted by EMBL. SSCP is based on a method which uses the protein's amino acid composition as input. This method is described in [11]. Several servers are available for analyzing protein secondary structures, and this particular server was chosen mainly because it was easy to work against using Perl. According to [11], the method used by this server is also very accurate. It is possible, however, that using another server would have led to better estimates of alpha helix and beta sheet content. Alpha helix and beta sheet content were not known in advance to correlate with nervous system specificity.

(11) *Presence of signal peptides*. Signal peptides occur in proteins that are secreted, i.e. released from the cell. Such proteins can perform messaging functions and occur widely in the nervous system. For instance, neuropeptide precursors contain signal peptides. A measure of the likelihood of each sequence being a signal peptide was obtained from the SignalP program [21] at CBS. The specific measure used was the "maximal Y score", which is a predictive indicator of whether a signal peptide is present in a protein or not.

(12) *Number of transmembrane helices*. Many neural proteins contain transmembrane helices: receptors, ion channels, pumps, etc. The TMHMM program [30] at CBS was used to predict the number of transmembrane helices. The resulting value was not divided by sequence length. The rationale was that the exact number of transmembrane regions may in fact be very meaningful for function and/or classification; for example, G-protein-coupled receptors always have seven transmembrane regions, no matter their size [1].

(13) *N-linked glycosylation sites*. N-linked glycosylation is thought to participate in a wide variety of processes, like protein folding, sorting and targeting [17]. The number of predicted N-linked glycosylation sites was obtained from ScanPROSITE [29]. However, N-glycosylation occurs on the luminal side of the endoplasmatic reticulum, and thus any protein that does not encounter this environment will not get N-glycosylated [14]. Therefore, all predictions for N-glycosylation sites were ignored for proteins that were predicted not to have a signal sequence (which is necessary for entry into the ER). It was not known whether N-linked glycosylation sites were expected to be more or less common in neural proteins compared to other proteins.

Note that some of the features are heavily dependent on each other, e.g., N-glycosylation site frequency and GlcNAc O-glycosylation frequency depend on the SignalP score.

3. MODEL CONSTRUCTION AND EVALUATION

In this section we first describe the original experimental setting, briefly describe the learning methods used, and then present the results obtained in the first experiment. We finally present results from using a variable elimination scheme.

3.1 Experimental setting

The null hypothesis for our first experiment is that no model can be induced that performs significantly better than randomly predicting the class labels (i.e., brain-specific or non-brain-specific protein). Since the size of the data set is limited (107 examples of each class), a v-fold cross-validation scheme was preferred to dividing the data once in a training and test set. The number of folds was set to 10.

3.2 Learning methods

We used Virtual Predict [5], which is an inductive logic programming system that is a successor of Spectre 3.0 [6]. The system can be viewed as an upgrade of standard decision tree and rule induction systems in that it allows for more expressive hypotheses to be generated and more expressive background knowledge (i.e., logic programs) to be incorporated in the induction process. The major design goal has been to achieve this upgrade in a way so that it should still be possible to emulate the standard techniques, with lower expressiveness but also lower computational cost. As a side effect, this has allowed the incorporation of several recent methods, such as bagging, boosting and randomization that have been developed for standard machine learning techniques into the more powerful framework of Virtual Predict.

Below, we briefly describe the methods and parameters that were used in the experiments.

3.2.1 *Decision tree induction using MDL*

A method for decision tree induction was defined by choosing the divide-and-conquer strategy together with the information gain criterion. An MDL criterion (a version of the criterion in [25] that has been modified to handle numerical attributes effectively) was chosen as a means to avoid over-fitting. This method was selected in favor of splitting the training data into a grow and a prune set, which due to the limited size of the data set would lead to highly variable decision trees.

3.2.2 *Bagging decision trees*

The divide-and-conquer strategy was also combined with bagging [4], which is a popular ensemble learning method for reducing variance. Following the scheme in [4], each decision tree in the ensemble is grown from a bootstrap replicate of the training set (i.e., a set created by randomly selecting, with replacement, n elements from the training set, where n is the size of the training set) and where the entire training set is used as a prune set. The number of trees in the ensemble was set to 50.

3.2.3 *Boosting decision trees*

Boosting is an ensemble learning method that uses a probability distribution over the training examples and re-adjusts the distribution after having generated a member of the ensemble. This is done in a way so that the learning algorithm focuses on those examples that are incorrectly classified by previous members. New examples are classified according to a weighted vote of the classifiers in the ensemble. The method used in Virtual Predict is called AdaBoost [13]. The base learning method was again the divide-and-conquer strategy using information gain and as well as a randomly selected prune set corresponding to 33% of the total weight. Again the number of trees in the ensemble was set to 50.

3.2.4 Randomizing decision trees

A third method for ensemble learning is to include a stochastic component in the learning algorithm and then let the stochastically generated members of the ensemble vote on new examples (e.g., [10]). The divide-and-conquer strategy was combined with the stochastic component in Virtual Predict, which selects a split with a probability that is proportional to the information gain of the split. Like for the two previous ensemble learning methods, the size of the ensemble was set to 50.

3.2.5 Decision lists

We also included a method that employs a totally different search strategy compared to the previous methods, namely separate-and-conquer. This strategy was combined with incremental reduced error pruning [9], where each clause immediately after its generation is pruned back to the best ancestor. The criterion for choosing the best ancestor is optional in Virtual Predict, and was set to the most accurate clause as estimated on training data using the m-estimate [8], with m set to 2. The system was instructed to treat the hypothesis as an ordered set (also often referred to as a decision list [27]).

3.3 Experimental results

In table 1, the results from the ten-fold cross-validation is shown. In addition to the overall accuracy, the mean number of rules generated by each model is also listed. This is because a biologist might prefer a model with slightly lower accuracy but only a few easily interpreted rules over a more accurate model with thousands of rules. According to McNemar's test, the difference in accuracy between the first four methods is insignificant (lowest p-value is 0.22), but all the first four methods are significantly more accurate than the fifth (at a 1% significance level). The two best methods, Boosting and Randomization, correctly classify 146 examples out of 214. The binomial tail probability of obtaining that many correct classifications by random guessing is 5.16 · 10-8. Since the significance level has to be divided by the number of methods tested (Bonferroni correction), this means that the null hypothesis can be rejected at the 0.000001 level.

Method	**Mean accuracy**	**Mean no. of rules**
Decision tree using MDL	64.49%	5.6
Bagging	66.82%	1287.5
Boosting	68.22%	927.8
Randomization	68.22%	3213.2
Decision list	53.27%	13.2

Table 1. The performance of the five methods.

3.4 An experiment with a variable elimination scheme

Since the number of examples is quite small relative to the number of attributes, we expected that variance could be reduced by removing some of the attributes (i.e., increasing the bias). In order to investigate whether the given set of background variables could be reduced, we inspected the trees produced in each fold by the first method, and realized that only three variables were used in a majority of the trees (SignalP and N-glycosylation, which were considered by all ten trees, and Aliphatic index, which was considered by nine trees). The remaining variables were used from zero to five times, and a new data set was generated by eliminating these ten attributes. We employed the first method (decision tree induction using MDL) to this new data set. Again we used ten-fold cross validation, which was repeated 10 times, and the mean accuracy of the method is presented in Table 2.

Method	**Mean accuracy**	**Mean no. of rules**
Decision tree using MDL	70.00%	4.26

Table 2. Results on reduced data set.

4. CONCLUSIONS

Our experiments have shown that it is possible to predict a protein's brain specificity to a reasonable degree by gathering information about low-level protein properties from on-line prediction servers and using this information to train classifiers. There were no significant differences between the classification performances of the four best methods used, although the fifth was clearly inferior. An interesting finding that requires further investigation is the fact that an improved level of accuracy can be achieved through elimination of some of the attributes – all except three. Further analysis is required in order to understand whether there is any biological explanation for the fact that these three attributes were retained.

The presented work indicates that it might be possible to construct a whole range of classifiers aimed at predicting various high level properties of proteins based on their low level properties. Examples of such high level properties that we plan to investigate in future work are: other classes of tissue specificity, protein localization, protein targeting etc.
Although this paper presents preliminary results, we believe that it can open up for new and interesting ways of exploring the world of protein function by using a distributed and collaborative approach and taking advantage of the vast amount of information that is becoming available on the Internet and through other sources.

5. REFERENCES

[1] Alberts, B., et al., “Molecular Biology of the Cell”, Garland Publishing (1994) 735

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

[3] Blom, N. Et al., ”NetPhos. Sequence- and Structure-Based Prediction of Eukaryotic Protein Phosphorylation Sites”, Journal of Molecular Biology 294(5), (1999) 1351-1362

[4] Breiman L., “Bagging predictors”, Machine Learning 24 (1996) 123-140

[5] Boström H., “Virtual Predict User’s Manual”, Virtual Genetics Laboratory 2001.

[6] Boström H. and Asker L., “Combining Divide-and-Conquer and Separate-and-Conquer for Efficient and Effective Rule Induction”, Proc. of the Ninth International Workshop on Inductive Logic Programming, LNAI Series 1634, Springer (1999) 33-43

[7] Boström H. and Idestam-Almquist P., “Induction of Logic Programs by Example-Guided Unfolding”, Journal of Logic Programming, Vol. 40 (2-3) (1999) 159-183

[8] Cestnik B. and Bratko I., “On estimating probabilities in tree pruning", Proc. of the Fifth European Working Session on Learning, Springer (1991) 151-163

[9] Cohen W. W., “Fast Effective Rule Induction”, Machine Learning: Proc. of the 12th International Conference, Morgan Kaufmann (1995) 115—123

[10] Dietterich, T. G., “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization”, Machine Learning 40, (2000) 139-158

[11] Eisenhaber F. et al., “Prediction of Secondary Structural Content of Proteins from Their Amino Acid Composition Alone. I. New Analytic Vector Decomposition Methods”, Proteins: Struct.,Funct.,Design, 25 N2, (1996) 157-168

[12] Fayyad U. and Irani K., “On the Handling of Continuous Valued Attributes in Decision Tree Generation", Machine Learning 8, (1992) 87-102

[13] Freund Y. and Schapire R. E., “Experiments with a new boosting algorithm”, Machine Learning: Proceedings of the Thirteenth International Conference (1996) 148-156

[14] Gavel Y. and von Heijne G., “Sequence differences between glycosylated and non-glycosylated Asn-X-Thr/Ser acceptor sites: implications for protein engineering”, Protein Engineering 3, (1990) 433-442

[15] Guruprasad K. et al., “Correlation between Stability of a Protein and its Di-peptide Composition: A Novel Approach for Predicting in vivo Stability of a Protein from its Primary Sequence”, Protein Eng. 4 (1990) 155-161

[16] Hansen, J.E. et al., “NetOglyc: Prediction of mucin type O-glycosylation sites based on sequence context and surface accessibility”, Glycoconjugate Journal 15, (1998) 115-130

[17] Helenius, A. and Aebi, M., “Intracellular Functions of N-linked Glycans”, Science 5512, (2001) 2364-2369

[18] Ikai A., “Thermostability and aliphatic index of globular proteins”, Journal of Biochemistry 88(6), (1980) 1895-8

[19] Kyte, J. and Doolittle, R.F., “A simple method for displaying the hydropathic character of a protein”, J Mol Biol 157, (1982) 105-32

[20] Lander, E.C. et al., ”Initial sequencing and analysis of the human genome”, Nature 6822, (2001) 860-921

[21] Nielsen, H et al., ”SignalP. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites”, Protein Engineering 10, (1997) 1-6

[22] ProtParam link: http://www.expasy.ch/tools/protparam.html

[23] Quinlan J.R., “Induction of decision trees", Machine Learning, 1 (1986) 81-106

[24] Quinlan J.R., “Learning logical definitions from relations", Machine Learning 5 (1990) 239-266

[25] Quinlan J.R and Rivest R.L, “Inferring Decision Trees Using the Minimum Description Length Principle”, Information and Computation 80(3) (1989) 227-248

[26] Revest, P. and Longstaff, A, “Molecular neuroscience”, Bios (1998) 31

[27] Rivest R.L., “Learning Decision Lists”, Machine Learning 2(3) (1987) 229-246

[28] Rudd, P.M. et al., “Glycosylation and the Immune System”, Science 5512, (2001) 2370-2376

[29] ScanProsite link: http://www.expasy.ch/tools/scnpsit1.html

[30] Sonnhammer, E.L.L et al., “A hidden Markov model for predicting transmembrane helices in protein sequences”, Proc. of Sixth Int. Conf. on Intelligent Systems for Molecular Biology, AAAI Press (1998) 175-182

[31] Venter, J.C. et al., “The Sequence of the Human Genome”, Science 5507, (2001) 1304-1351

[32] Wells, L. et al., “Glycosylation of Nucleocytoplasmic Proteins: Signal Transduction and O-GlcNAc”, Science 5512, (2001) 2376-2378