Scoring protein function predictions

Björn Pollex

April 19, 2010

1 Introduction

Due to advances in genome sequencing technology, more and more fully sequenced genomes are becoming available at an ever accelerating rate. However, the relationship between genomic sequence and biological function is complex, and the key to understanding it, are the gene products [11]. One important class of gene products are the proteins. Databases like the UniProtKB (`www.uniprot.org`) strive to archive all the available information about proteins, such as their amino acid sequence and annotations that describe their function. They use the structured vocabulary of the Gene Ontology [1] to describe gene products in terms of their molecular function, biological process and cellular location. Since these databases are still incomplete [12], and experimentally determining the function of uncharacterized proteins is very expensive, prediction algorithms are used to infer new annotations for poorly and unannotated proteins from the existing knowledge base ([3], [8]). These inferred annotations can then be used as a hypothesis for further work. Based on the assumption that such an algorithm might have more supporting evidence for some predictions than for others, it is logical to conclude that some predictions are more likely to be correct than others. Since manual verification of such predictions can be a very time-consuming process, it is useful to be able to differentiate between them. The different levels of confidence an algorithm has in its predictions can be expressed by confidence scores.

2 Aim

In this thesis, we plan to enhance an existing algorithm for protein function prediction proposed in [6] with confidence scoring. We will use a Support Vector Machine (SVM) trained of predictions made by the original algorithm to decide, given the evidences provided by the algorithm, whether a prediction is true or false, and then use the distance value calculated by the SVM to obtain a confidence score for the prediction. In its current form, the algorithm employs various manually calibrated thresholds in order to make binary decisions based on continuous features. Now the SVM can learn these thresholds for us, and we hope that this will increase the coverage of the algorithm.

The section *Background* will give a brief description of the protein function prediction algorithm this thesis is based on. In *Problems and Goals* we will discuss the challenges in more detail, and define criteria for assessing success or failure of our approach. *Methods* will describe in more detail how we plan to overcome these challenges and finally *Expected Results* will give a short overview of what we expect to find.

3 Background

The function prediction algorithm this thesis is based on follows a "guilt by association"-approach, where new annotations for a protein are transferred from proteins that are assumed to have a similar function. There are two main evidences that indicate that two proteins may have similar functions:

Orthology Evolutionary related species have proteins that did not change much after these species separated (e.g. hemoglobin in vertebrates). Often such proteins are well studied in model organisms such as E.Coli or yeast, but not in others, such as human, where function is more difficult to study.

Neighborhood/Module Based on information of physical interactions between proteins, one can construct a Protein-Protein-Interaction (PPI) network [2]. The nodes in such a network represent proteins, and edges represent interactions. Protein complexes like the ribosome form densely connected clusters, so-called functional modules, in such a network, and proteins within the same module are more likely to have similar functions.

The algorithm compares the PPI networks of two or more species and finds subgraphs of orthologous proteins with conserved interactions. These subgraphs are called *conserved and connected subgraphs* (CCS).

The algorithm then iterates over all proteins that are members of a CCS, and checks whether it is possible to transfer annotations from orthologs and/or neighbors to weakly or non-annotated proteins.

In a more abstract view one could say that the algorithm finds potential protein-annotation-pairs and then uses a binary classifier to decide if these are valid predictions.

4 Problems and Goals

In its current form, this algorithm outputs a potentially very long (about 12000 for the human proteome), unordered list of predictions. The main goal of this thesis is to enhance the algorithm to produce confidence scores for the predictions it makes.

Let a prediction be a tuple

$$\text{pred} = (P, A)$$

where P denotes a protein and A a functional annotation predicted for that protein. Let $\mathbb{P}$ be a set of predictions made by some method on some set of data. Suppose we are a curator of a database for functional annotations, and we want to verify these predictions in order to add the new annotations to our database. After tediously verifying the first 100 predictions, we have x true positives. Could we have done better? Laking any sensible criterion we had to pick predictions for verification at random. If we assume that there is some pattern that distinguishes correct prediction from incorrect ones based on the information we have available, we could try to enhance the prediction method to give us predictions of the form

$$\text{pred}_{\text{scored}} = (P, A, S)$$

where P and A are as before, and S denotes the confidence score assigned to that prediction. Candidates for verification can then be selected based on their confidence score, starting with the highest. To evaluate the scoring scheme, we again count the number x_{scored} of true positives after 100 verifications, and determine how much it has increased. Let

$$\Delta x = x_{\text{scored}} - x$$

be the increase of true positives. We have achieved an improvement over our first approach, if $\Delta x > 0$. One goal of this thesis is to enhance the algorithm proposed by [6] to output confidence scores for the predictions it makes, such that Δx is maximized.

As mentioned in *Background*, at its core the algorithm can also be modelled as a binary classifier that takes a protein P and a functional annotation A as it's input, and then decides whether the given protein has the given annotation. When determining if a function might pertain to a protein, it checks several criteria. One of these is for instance:

Is the functional similarity of P and its direct neighbor P' greater than f?

The total number of similar neighbors that support the function in question is later used to decide, whether the annotation should be transferred. By discarding all neighbors that are not similar enough, information is lost. The underlying assumption is, that only similar neighbors are likely to have the same function. However, a very great number of less similar neighbors might also justify a prediction, but that information is lost by using the threshold f. An algorithm that assigns confidence values to it's predictions does not have to make a strictly binary decision anymore, instead it can use the available evidence to calculate the confidence score for every potential prediction. This way, predictions that would have been discarded at an early stage before are now considered throughout all stages of the algorithm and may yield a high overall score in the end, and hopefully prove to be true positives. Therefore, this thesis will also study how the removal of thresholds affects the overall performance of the algorithm.

5 Approach

5.1 Theory

We will use a machine-learning approach, specifically a support vector machine (SVM), to achieve the goals stated above, as these have been used in similar scenarios with good results (see [14], [7]).

Let $F_1, \ldots, F_n$ be a set of features, and $F_1(p, a), \ldots, F_n(p, a)$ be the measured values of these features for protein P and annotation A. These features are the evidence we have to decide whether A pertains to P. In our scenario possible features may include, but are not limited to:

- The number of neighbors of P with annotation A.
- The average similarity of these neighbors.
- The number of orthologs of P with annotation A.
- The average similarity of these orthologs.
- The functional coherence of an orthologous group.
- The functional coherence of the underlying CCS.

The choice of features is somewhat predetermined by the current implementation of the algorithm, but there are is some room for new features such as weighting the interactions.

The concrete binary classification problem we want to solve with an SVM can be formulated as follows:

Given $F_1(p, a), \ldots, F_n(p, a)$, does A pertain to the protein P?

Asking such a classifier about every possible protein-annotation-pair would be unfeasible, so the search space has to be narrowed down. As in the original algorithm, predictions will only be evaluated for proteins that are part of a CCS. Potential annotations for a protein are those that are represented at least once among its neighbors or orthologs, as other annotations would never have any supporting evidence.

An SVM needs a representative set of training-data in order to learn how to separate the two classes (in our cases true and false predictions). Although the specific GO-term and protein are variables of the evidence pattern for a prediction, we still have to make sure, that every existing evidence pattern is represented in the training-data. Since we do not have training-data for every possible protein-annotation-pair, we will try to generalize evidence-patterns into classes. After transferring all of our known annotation into the feature space by extracting their evidences, we will use a hierarchical clustering algorithm to partition the data-points into clusters. Each of these clusters represents an evidence pattern class. We will then count the distribution of each protein-GO-term-pair among the clusters. Proteins will not be counted directly, but rather by their class (e.g. SCOP[9] or CATH[10] classes). From these counts we will obtain a matrix of weights for each cluster, containing the weight of that cluster for each protein(class)-GO-term-pair. The sum of the weight-matrices of all the clusters will be the unity-matrix. We will then train a separate SVM for each of the clusters. When querying for a new annotation later, the result could be the weighted average of each of the individual SVMs.

In order for an SVM to be able to handle a certain data-type, it needs a kernel-function for that data-type. A kernel-function assigns a similarity-value to any two objects of the same data-type. Kernel-functions for complex data-types are usually linear combinations of the kernel-functions of the primitive date-types that make up the complex one (see [7]). This thesis will have to deal with the problem of choosing and combining appropriate kernel-functions for all the input-features.

An SVM needs positive and negative training samples in order to construct a separating hyperplane between them. As positive training sample we can use protein-annotation-pairs that were correctly recovered by the original algorithm after blanking them out. Positive samples will only be chosen from within the cluster that an SVM is training for. Current databases contain almost no negative data, that is, information on which annotations a protein is known *not* to have. Nevertheless, learning classifiers from an incomplete gold-standard has been shown to be practical in spite of that problem [5]. In order to overcome this, we will have to define a sensible criterion for true negatives. In general, we will assume that proteins that do not have a certain annotation are true negatives for that annotation. That way we penalize the algorithm for every novel annotation it predicts, thus introducing a heavy bias into our training data. In order to mitigate that bias, we can refine this criterion in several ways, e.g.:

- only classify novel annotations for already annotated proteins as negatives
- classify annotations as negatives based on their impact on the functional coherence of the annotations of that protein

These criteria will have to be carefully balanced in order to obtain a suitable set of training data.

There are several approaches to extract confidence scores from an SVM. One approach is to take the distance of a vector from the separating hyperplane in the feature space as a basis. However, when normalized, this value does not necessarily correlate with the posterior class membership probability. [13] compares several methods for calibrating the output of an SVM. Another way to obtain probability estimates is to use a voting approach as shown in [14] (this is interesting because we do train several SVMs). Whether any calibration is necessary, or the distances given by the SVM are sufficient as confidence scores, will have to be determined in this thesis.

5.2 Implementation

As mentioned above, we intend to remove some thresholds from the current implementation of the algorithm, hoping to increase coverage. In order to extract all the features needed for the SVM, extensive modifications to the current implementation are necessary. Due to the extend of these changes, it is likely easier to re-implement the algorithm altogether. This will be done using a combination of C++ and Python. Several implementations of support vector machines are freely available (e.g. libSVM, [4]).

For reasons of complexity and performance, we will not change anything about the process of orthology-detection and the constructing of the CCS. Subjecting these steps to a learning algorithm would require us to consider an unfeasibly large number of possible CCS. We will therefore consider the CCS and orthologous groups as given.

The performance of an SVM classifier depends heavily on the training data. Following the path taken in [6] we will train and evaluate SVMs for different species combinations.

6 Expected Results

By introducing confidence scores, we hope to increase the precision of the predictions, when evaluating only the best scored predictions, while with the removal of thresholds that filter out possible predictions we intend to increase coverage.

An advantage of using an SVM for classification is, that by studying the hyperplane the SVM learns, we may be able to better understand the influence of certain features on the score. For instance, one could fix the values for all features except the number of supporting neighbors, and then study how the number of true positives depends on that. This in turn can help us to further refine the features used for making predictions.

References

[1] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[2] A. Barabási and Z. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[3] X. Chen, M. Liu, and R. Ward. Protein function assignment through mining cross-species protein-protein interactions. *PLoS ONE*, 3(2), 2008.

[4] C. Chih-Chung and L. Chih-Jen. *LIBSVM: a Library for Support Vector Machines*, 2006.

[5] C. Huttenhower, M. Hibbs, C. Myers, A. Caudy, D. Hess, and O. Troyanskaya. The impact of incomplete knowledge on evaluation: an experimental benchmark for protein function prediction. *Bioinformatics*, 25(18):2404, 2009.

[6] S. Jaeger and U. Leser. Combining modularity, conservation, and interactions of proteins significantly increases precision and coverage of protein function prediction. Draft, 2010.

[7] D. P. Lewis, T. Jebara, and W. S. Noble. Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. *Bioinformatics*, 22(22):2753–2760, 2006.

[8] R. Llewellyn and D. Eisenberg. Annotating proteins with generalized functional linkages. *Proceedings of the National Academy of Sciences*, 105(46):17700, 2008.

[9] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[10] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, and J. Thornton. CATH-a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[11] A. Pandey and M. Mann. Proteomics to study genes and genomes. *Nature*, 405(6788):837–846, 2000.

[12] S. Rhee, V. Wood, K. Dolinski, and S. Draghici. Use and misuse of the gene ontology annotations. *Nat. Rev. Genet*, 9(7):509–515, 2008.

[13] S. Rüping. A simple method for estimating conditional probabilities for svms. Technical report, Dortmund University, 2004.

[14] A. Vinayagam et al. Applying support vector machines for gene ontology based gene function prediction. *BMC Bioinformatics*, 2004.