Proteome Analyst – Transparent High-throughput Protein Annotation: Function, Localization and Custom Predictors

D. Szafron[1], P. Lu[1], R. Greiner[1], D. Wishart[2], Z. Lu[1], B. Poulin[1], R. Eisner[1], J. Anvik[1], C. Macdonell[1] and B. Habibi-Nazhad[2]

[1]Department of Computing Science University of Alberta, Edmonton, AB, Canada, T6G 2E8 and [2]Faculty of Pharmacy and Pharmaceutical Sciences, University of Alberta, Edmonton, AB, Canada, T6G 2N8

ABSTRACT

Motivation: Modern sequencing technology now permits the sequencing of entire genomes, leading to thousands of new gene sequences in need of detailed annotation. It is too time consuming to predict the properties of each protein sequence manually and to organize the results of many prediction tools by hand. The prediction process must be automated so the predictions can be automatically organized, but the predictions must also be transparent. That is, the rationale for each prediction should be easily examinable by anyone that wishes to use the prediction.

Results: Proteome Analyst (PA) is a web-based system for predicting the properties of each protein in a proteome. PA has three interesting features. First, it provides a single web-based tool that allows the user to select a wide range of analytic tools and automatically apply them to each protein in a proteome. In essence, PA provides *one-stop automatic high-throughput analysis*. Second, PA has the ability to explain its predictions to users. PA is based on established machine learning techniques, but makes every prediction transparent to its users. Third, PA allows users to easily create their own transparent custom predictors without programming.

Availability: http://www.cs.ualberta.ca/~bioinfo/PA

Supplementary Information:
http://www.cs.ualberta.ca/~bioinfo/PA/Walkthrough
http://www.cs.ualberta.ca/~bioinfo/PA/Experiments

Contact: bioinfo@cs.ualberta.ca

INTRODUCTION

High-throughput sequencing technology has made it possible for even the smallest laboratory to sequence the genome of organelles, viruses and bacteria. Indeed there are now more than 1200 genome sequences deposited in public databases (http://www.ebi.ac.uk/genomes/) and this number is growing rapidly. Given the size and complexity of these data sets, most researchers are compelled to use automated annotation systems to identify or classify individual genes and proteins in their genomic data. A number of excellent systems have been developed over the past few years that permit automated genome-wide or proteome-wide annotation. These include GeneQuiz (Andrade *et al.*, 1999), GeneAtlas (Kitson *et al.*, 2002), EnsEMBL (Hubbard *et al.*, 2002), PEDANT (Frishman *et al.*, 2001), Genotator (Harris, 1997), Magpie (Gaasterland and Sensen, 1996) and GAIA (Overton *et al.*, 1998). These use web-based tools to identify genes, parse data, translate sequences, search against public databases, identify domains or motifs and perform predictive analyses of protein sequences. Many of these packages provide user-customizable search schemas and graphical, hyperlinked output. The level of interpretation or inference offered by these annotation systems varies widely, with some offering only raw data (lists of homologues, calculated properties, etc.) in a consolidated format and others inferring function or ontology through detailed lexical analysis.

Our PA system focuses on the task of prediction or classification. Our results show that classification/prediction can be used for many annotations. The most obvious is general-function, but we are also working on sub-cellular localization, specific function and protein-protein interaction are others we are working on.

To use classification, we require an ontology or "dictionary" of class terms. Consequently, a number of controlled vocabularies or ontologies of protein function have started appearing. Among the first was the enzyme classification scheme (E.C. number) developed by the IUBMB and first employed in the ENZYME database (Bairoch, 1993). Later, Riley (1993) developed functional classification schemas for E. coli and other microbes. Variations on these functional vocabularies were added as other genomes and genome annotation systems were

published or developed (Blattner *et al.*, 1997; Andrade *et al.*, 1999). More recently, these ontologies have become somewhat more consolidated with the establishment of the Gene Ontology Consortium (Ashburner *et al.*, 2000). The Gene Ontology (GO) consortium uses a controlled vocabulary to provide functional, process and component classification of proteins through a dictionary of about 9,125 terms. However the GO vocabulary may not be suitable for all applications or in all situations. Obviously some choice or vocabulary "customization" would help.

In addition to the choice of the controlled vocabulary, the rules of protein/gene assignment or annotation have to be sufficiently well defined to permit computer automation and to support transparent explanation. A number of approaches that incorporate elements of text processing (word counting) and artificial intelligence (neural nets, hidden Markov models) have been developed and used in automated genome annotation systems (Raychaudhuri *et al.*, 2002; Xie *et al.*, 2002; Schug *et al.*, 2002; Gaasterland and Sensen, 1996). However, these approaches do not necessarily lend themselves to "expert" validation or transparent explanation. The decision process is, therefore, effectively hidden. We have developed a web-accessible program called Proteome Analyst (PA), to address these issues of:

1) customizable or flexible ontologies,
2) web-accessible automation and
3) transparency in proteome annotation,

PA makes extensive use of machine learning (ML) techniques. This paper focuses on ML techniques that build classifiers to make predictions. In general, a classifier predicts one of a fixed number of classes for each query item (Mitchell 1997). Here, the classes are from a biological ontology and the query items are protein sequences.

A classifier must be built before it can be used. A common technique is to apply a known ML algorithm to a set of labeled training items to produce a classifier. In our case, each training item consists of a primary protein sequence and the ontological class it has been assigned by an expert. For example, the GeneQuiz (GQ) web-site (http://jura.ebi.ac.uk:8765/ext-genequiz) has a list of sequences from many organisms and their respective GQ ontological classes. Therefore, a training set can be taken directly from this site. Alternately, the SWISS-PROT database (Bairoch and Apweiler 1997) contains PIR numbers for many sequences and the GO web-site (www.geneontology.org) provides tables to map PIR numbers to their ontological classes. These can be used to construct a training set. In this paper, we use GQ data for training our general-function classifiers.

In general, a ML classifier algorithm requires features to be associated with each training item. The existence or absence of the features in each training item, along with the known class for each item, are used to build a classifier. In PA, a feature is a word or a phrase found in the SWISS-PROT database entries of homologues to the protein in question. Features are not provided in the training set – they are computed automatically.

Once built, a classifier takes a list of sequences with unknown classes and uses the existence or absence of features to predict their classes.

In addition to pre-built classifiers, we also show how PA can be used to create custom classifiers. For example, Nair and Rost (2002) used a classifier to predict the localization of proteins in the cell. We show how PA can be used by a molecular biologist with no programming experience to quickly train a custom classifier that improves their results. As another example, we train a sample custom classifier for K^+-Ion channel proteins (Gallin and Spencer 2001).

In the context of PA, *transparency* is the ability to provide formally sound and intuitively simple reasons for each prediction. PA makes its decisions based on well-understood concepts of conditional probabilities. Its explanations use ordered lists of anomalies and stacked bar-graphs (Fig. 9) that clearly display the evidence for each prediction.

Although there are a variety of tools for protein annotation, PA makes the following scientific contributions: First, PA provides a single, integrated, high-throughput and web-based interface to a number of different analytical tools. Entire proteomes can be analyzed according to (currently) 33 properties, including protein function and sub-cellular localization, without the need for human intervention. Second, PA allows the user to create custom predictors in a simple train-by-example way. Third, provides clear and transparent explanations for each of its predictions in a novel and intuitive manner.

SYSTEMS AND METHODS

Proteome Analyst is web-based. The user may choose to either analyze a proteome or train a custom classification-based predictor. In this section, we explain both.

Analysis of a Proteome

To analyze a proteome, the user must first upload a FASTA format file containing the sequences to be analyzed. The user only needs to provide a name for the proteome, and to navigate to the location of the file to be uploaded.

After submitting the proteome, PA uploads the proteome file from the user's file system, extracts the sequences, and stores them in a MySQL database (http://www.mysql.com). The user can either commence the analysis, using a default selection of tools with their respective default settings, or choose to select the analysis tools and specify their respective configuration parameter values.

PSI-BLAST (Altschul *et al.*) is the only tool that is always run during an analysis. It is used to find homologues. The user may change various parameters of PSI-BLAST such as

the number of iterations, the database in which to search for homologues, or the scoring matrix used. Other tools, like PSIPRED (Jones 1999) for doing secondary structure prediction, are optional. Optional tools can be configured to run only under certain conditions – *e.g.* only if the best homologue for a sequence is within a certain E-value range.

Two important tools are a classifier-based predictor and PACardCreator. Currently, a user may select from several pre-built general-function classifiers that use the GQ ontology and were trained on sequences from individual organisms: E.Coli, Yeast, Drosophila or a multi-organism trained classifier trained with sequences from all three organisms. Alternately, a user may select any custom classification-based predictor that has been trained as described in the next sub-section.

The PACardCreator generates a PACard for each sequence, which is a convenient summary of all the predicted properties of each protein in a proteome. The top of a typical PACard is shown in Fig. 1. A PACard is based on the E.Coli cards from the CyberCell Database (CCDB) (http://redpoll.pharmacy.ualberta.ca /~bahram/CCDB.html).

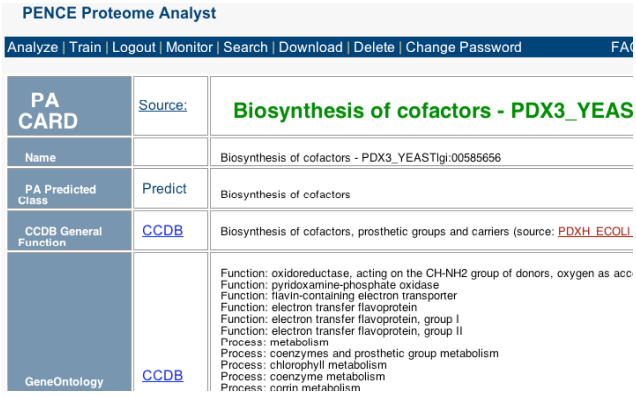


Fig. 1. The top part of a sample PACard.

Currently PA fills in up to 33 different fields: Name, PA Predicted Class, GeneOntology, Specific Function, Pfam Domain/Function, EC Number, Specific Reaction, General Reaction, PROSITE Motifs (Falquet *et al.* 2002), PSI-BLAST Results, Important Sites, Inhibitor, Interacting Partners, Sequence, Secondary Structure, Location, Metabolic Importance, Copy Number, RNA Copy No., Similarity, Number Of Amino Acids, Molecular Weight, Transmembrane, Cys/Met Content, Structure Class, Quaternary Structure, Cofactors, Metals Ions, Kcat Value (1/min), Specific Activity (micromol) and Km Value (mM). The underlined fields are computed and the others are predicted. A field is left blank if no prediction can be made.

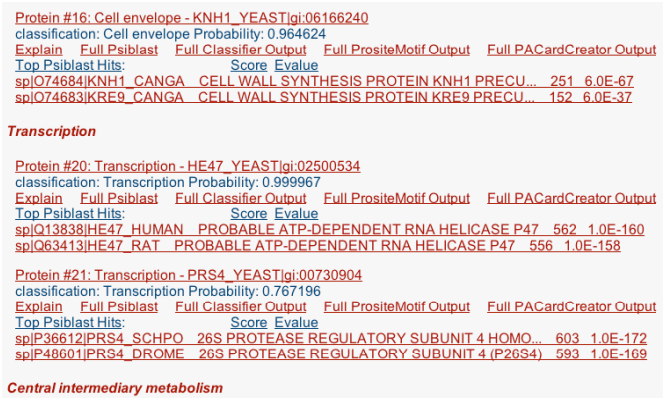
The tools are run in an assembly line fashion with the results of one tool being fed into the next tool. First, PA attempts to find homologues in the SWISS-PROT database using PSI-BLAST. In general, for most of the PACard fields, we simply fill in the value taken from the associated field of the SWISS-PROT database entry for the top homologue. In ML terminology, we are using a 1-nearest-neighbor approach as our default predictor for fields. However, if one of the top homologues is a sequence from E.coli, then we use the CCDB entry of this homologue (instead of the SWISS-PROT entry) to predict many of the fields in the PACard. This is because E.coli is one of the most accurately annotated proteomes and is well summarized in the CCDB.

We use a more sophisticated approach to predict the values for some of the fields. In particular, the production-version of PA currently uses a classifier to predict the general-function of each protein using the GQ ontology and the beta version of PA uses a classifier for the location (sub-cellular localization) field. As PA evolves, we will replace many of the other simple predictors with more sophisticated ones. The rest of this paper focuses on such predictors.

Fig. 2. shows the analysis results by ontology class, displaying the ontological class predicted for this instance, together with the probability that this instance belongs to the class. In addition, it also gives the top two homologues found during the PSI-BLAST search and provides a link to the full PSI-BLAST output in standard format. It also gives a link to the full Classifier output (Fig. 3), the PACard (Fig. 1) and any other tools that the user requested. In this case, there is a link to a list of PROSITE Motifs that PA found in each sequence.

Fig. 2. Proteins by ontological class.

Fig. 3 shows that the predicted ontological class of this sequence, *Transcription*, has a probability of only 76.7%. Other possibilities are *Energy metabolism* at 20.6%, or *Replication* at 2.7%. A link is also provided to an explanation of how the prediction probabilities were computed. This *Explain* facility is discussed later in the paper and is one of the most important characteristics of PA. We believe that it is essential for the widespread acceptance of computational prediction techniques in bioinformatics.



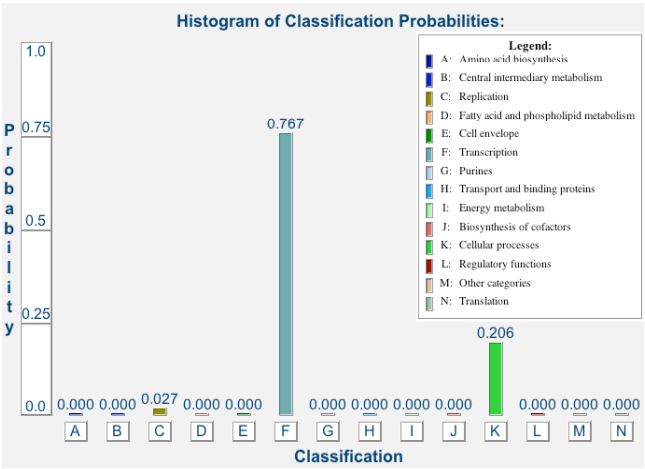


Fig. 3. The full classifier output for PRS4_Yeast.

Training a Custom Classifier

The production-version of PA includes several classifiers that have been trained to predict general-function. However, a user can also train custom classifiers without doing any programming or analysis, since PA can learn by example. The first step in training a custom classifier is to provide a name for the classifier and a corresponding training file in FASTA format. Each sequence in the file must have a FASTA tag that starts with a known class label. For example, Fig. 4 shows part of a training file for a custom K-ion channel classifier, where the two training sequences have known class labels KV1 and KV2 respectively.

```
>Kv1<VIC0 potassium voltage-gated channel,
shaker-related subfamily, member 2 [Homo
sapiens].
mtvatgdpadeaaalpghpqdtydpeadheccervvinis...
>Kv2<VIC171 potassium channel protein shab11
- fruit fly (Drosophila melanogaster).
mvgqlqgggaagqqqqqqqqatqqqqhskqqlqqqqqqqqq...
```

Fig. 4. The format of a classifier training file.

Once the classifier has been trained, the user may view the classifier information page shown in Fig. 5. It contains a summary of the training results. This includes three lists. The first list shows the training sequences that PA had to exclude since the PSI-BLAST on them did not produce any usable features. The second list contains training sequences that are most likely labeled incorrectly, sorted from the highest to the lowest probability of being mislabeled. The third list contains the rest of the training sequences, sorted from the highest to the lowest probability of being labeled correctly. Similar to analysis results, the user may ask why PA infers that a training sequence was labeled correctly or incorrectly by selecting an *Explain* hyperlink or looking at the raw PSI-BLAST results.

Fig. 5. Classifier training results information

ALGORITHMS

As shown in Fig. 6, classification-based prediction is a two-step process: training/learning and prediction. In the training/learning step, a classifier is built using a ML classification algorithm by analyzing a set of training sequences, each tagged by a known class label. In the prediction step, the generated classifier is used to predict the class label of an unknown query sequence.

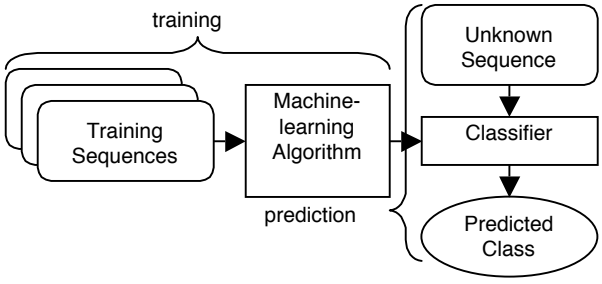


Fig. 6. The training and predicting phases of classification.

To create and use a classifier, PA uses a pre-processing step that maps each sequence to a set of features, as shown in Fig. 7. The sequence is first PSI-BLASTed against the SWISS-PROT database. The SWISS-PROT entry of each of the top three homologues (whose E-value is less than 0.001) is parsed to extract a feature set (described below). The union of the features for all three homologues forms the feature set for the original sequence. If no homologues match the E-value cutoff or if all features are removed by feature selection (see the next sub-section) then the sequence will have no features, so no prediction will be made.

The features we use are based on the SWISS-PROT KEYWORDS field and any Interpro numbers (Apweiler 2001) contained in the DBSOURCE field. We tried three different techniques for extracting features from the KEYWORDS field and compared the accuracies of the

resulting classifiers. The *full phrase* approach used each semi-colon delimited phrase as a single feature. The *simple token* approach used all individual words as separate features and the *stemming* approach used stemmed individual words as features. For example, Fig. 8 shows part of the SWISS-PROT entry for a homologue with the features used by PA in **boldface**.

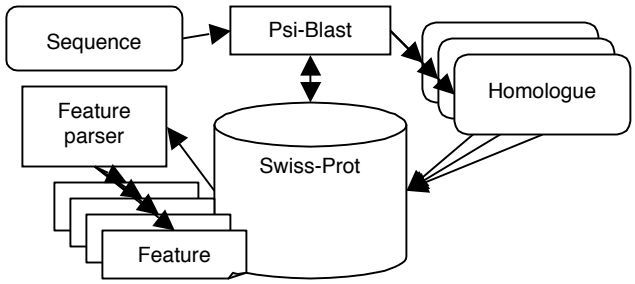


Fig. 7. Computing features for a protein sequence.

```
DBSOURCE  swissprot: locus MPPB_NEUCR, ...
  xrefs (non-sequence databases): ...
  InterPro IPR001431,...
KEYWORDS  Hydrolase; Metalloprotease; Zinc;
  Mitochondrion; Transit peptide;
  Oxidoreductase; Electron transport;
  Respiratory chain.
```

Fig. 8. Extracting features from a SWISS-PROT entry.

The *full phrase* approach would produce 9 features. The simple token approach produces 12 features. The 6 single word tokens are the same, but the 3 double word features are split into 6 single word features. The stemming approach (Sahami 1999) extends the simple token approach by stemming each feature. For example, if another SWISS-PROT entry contained the feature *transporter*, it would be stemmed to *transport* so that the features would match. The *full phrase* approach produced the most accurate classifiers, so we use it in PA.

Feature Selection – Wrappers

Unfortunately, PA extracts thousands of features from the SWISS-PROT database and standard ML algorithms do not scale well with a large number of features. Some features, like *Complete proteome* and *3D-structure* appear very often, and so have little predictive value. Therefore, they are put into a list of stop-words that are immediately filtered from the feature sets during training.

Since manually selecting or filtering features is tedious and error-prone, we investigated an automatic feature selection technique. The information content [Mitchell 1997] is computed for each feature generated by the set of training sequences. Informally, the information content of a feature is a measure of how well that feature differentiates between different classes in the ontology. For example, a feature that occurs in every training sequence of every class has information content 0, while a feature that occurs in every training sequence of a single class, but no other classes, has high information content.

We use a wrapper (Kohavi and John 1997) to select the subset of all the features, whose information content is above a threshold. When PA trains a classifier, it actually trains a series of classifiers with different wrapper thresholds. The first classifier uses all of the features. Each subsequent classifier is trained after removing an additional 5% of the features that have the lowest information content (Mitchell 1977). After the 20th round, only 5% of the features are left. We use 5-fold cross-validation (described in the next sub-section) to compare the classification accuracies of these 20 rounds and choose the *information content threshold* that produces the highest prediction accuracy.

Classifier Techniques

We tried four different ML techniques in PA: Naïve Bayes (NB) (Mitchell 1997), Tree-augmented Naïve Bayes (TAN) (Friedman *et al.* 1997), Artificial Neural Network (ANN) (Mitchell 1997) and Support Vector Machines (SVM) (Burges 1998). The simplest of these techniques is NB. Given a description for an instance, the NB classifier returns the probability that this instance belongs to each possible class, P_i, for each of the k classes, $C_1 \ldots C_k$.:

$$P_i = P\left(C = C_i \mid {}^{+}F_1, {}^{-}F_2, \ldots, {}^{+}F_m\right)$$

where ${}^{+}F_j$ and ${}^{-}F_j$, respectively denote that the j-th feature appears or doesn't appear in the description. Using Bayes rule and the (very strong) assumption that features are independent, given the class, this corresponds to:

$$P_i = \frac{({}^{+}c_{i1} \cdot {}^{-}c_{i2} \cdot \ldots \cdot {}^{+}c_{im})}{K * n * (n_i)^{m-1}}$$

where ${}^{+}c_{ij}$ and ${}^{-}c_{ij}$ respectively denote the number of training instances with class label C_i that include and do not include the feature F_j, the total number of training instances with class label C_i is denoted n_i and the total number of training instances is denoted n. K is a normalization constant to make the sum of the probabilities *1*. We also use a Laplacian correction (Mitchell 1997) to avoid indeterminate forms: *0/0* as explained in a later section.

TAN's are similar to NB's, but can include a limited range of dependencies between pairs of features. We use a three-level ANN, with the number of input nodes equal to the number of features, a single layer of hidden units and a number of output units equal to the number of ontology classes. We used 10 hidden units, but the experiments showed that varying this number had little effect. Training consisted of 100 cycles through the training set. A simple SVM learns how to classify objects into two classes using hyper-planes that separate the training data by a maximal margin. We used the linear SVM kernel (Burges 1998).

We compare the accuracy of classifiers using 5-fold cross validation (Mitchell 1997). The training sequences are partitioned into five sets. A classifier is constructed using sequences from four of the five sets as training sequences and applying the resulting classifier to all sequences in the fifth (test) set. The accuracy of the classifier is the percentage of sequences in the test set whose class was correctly predicted. Five classifiers are constructed, each using a different one of the five sub-sets as the test set. The 5-fold cross-validation accuracy is the average of the five accuracies computed.

EXPERIMENTS

General-Function Prediction

We began by building a series of classifiers to predict the general-function of proteins, based on the GQ ontology. Each classifier was trained using training data from a single organism obtained from the GQ web-site (http://jura.ebi.ac.uk:8765/ext-genequiz). We selected three organisms with highly-annotated sequences in the SWISS-PROT database: E.coli (2370 sequences), Yeast (2359 sequences) and Fly (Drosophila melanogaster) (3842 sequences).

We began by evaluating the effects of wrapping. For example, the unwrapped Naïve Bayes classifier had an accuracy of 82.4%. However, the accuracy changes as a function of the percentage of features that were filtered by a wrapper. The best accuracy of 82.5% was achieved by removing 15% of the features and the accuracy was still at the original 82.4% level if 55% of the features are removed. Each of the ML techniques had different improvements to prediction accuracy when wrappers were used. For example, the TAN classifier improved from 67% to 77.2% on E.coli. Even in cases where the wrapper did not significantly increase accuracy, it is still very helpful, since removing many features has two advantages. First, it produces faster classifiers, which is important in the high-throughput domain. Second, it simplifies the prediction explanation process, which is crucial to PA.

Table 1 compares the best accuracy (after wrapping) of the four ML techniques described in the previous section. ANNs and SVMs have the best prediction accuracy, by about 10%.

Table 1. Accuracy of four classifier algorithms with wrappers

	NB	TAN	ANN	SVM
E.coli	82.5%	77.2.4%	93.1%	87.1%
Yeast	78.8%	74.0%	95.2%	85.1%
Fly	76.6%	72.9%	91.8%	81.1%

We then trained several custom classifiers to determine whether the same ML techniques can be used to accurately predict PACard fields other than general-function.

K^+-Ion Channel Proteins

We performed a case study to determine how well our four wrapped ML algorithms would fare when each sequence had a large number of homologues across several different ontological classes. We wanted to determine if our techniques could automatically sift through the features and identify the right features to make accurate predictions.

Voltage-gated potassium channels (VKC) are intrinsic membrane proteins that respond to changes in transmembrane electric field by changing shape and selectively allowing potassium ions to pass through the lipid bi-layer (Gallin and Spencer 2001). We obtained 78 protein sequences that were divided into four classes (KV1 – 23 sequences, KV2 – 19 sequences, KV3 – 17 sequences and KV4 – 19 sequences) from W. Gallin's lab at the University of Alberta. Many of the VKC sequences have close homologues that lie in classes other than their own class.

PA produced an NB classifier that initially made only 3 errors during 5-fold cross validation. However, one error was a labeling error in the training set. We eliminated the other two errors by modifying the feature extraction algorithm. Originally, we performed three PSI-BLAST iterations before picking the top 3 homologues to use for feature extraction. We found that when there are many homologues in different ontological classes, better accuracy can be obtained by using only a single PSI-BLAST iteration (which is equivalent to a Blast-P computation).

The reason is that multiple PSI-BLAST iterations tend to promote sequences from the most prevalent organisms in the SWISS-PROT database, at the expense of sequences from minority organisms, even though the minority sequences may have better similarity. The lesson here is that when you train a classifier for predicting properties that differentiate based on small differences, a single iteration is better. After making this change, the cross-validation accuracy of all four classifiers increased to 100% on the K-Ion training set.

Sub-cellular Localization

In the 2002 ISMB conference, Nair and Rost (2002) presented some interesting results on predicting sub-cellular localization, using keywords from SWISS-PROT. We viewed their results as another good opportunity to test how well our ML techniques can be applied to prediction tasks other than general-function prediction. They shared their training data (3146 sequences) and ontology (10 classes) with us and we constructed a PA custom classifier, using only their raw primary sequence data and sub-cellular class labels. We did not do any manual feature identification. We also used their full set of sequences, instead of the smaller set of 1146 sequences that produced their good prediction results (81.5% accuracy on 36.9% coverage). PA automatically produced a wrapped NB classifier with 87.8% accuracy on 100% coverage, using 5-fold cross validation.

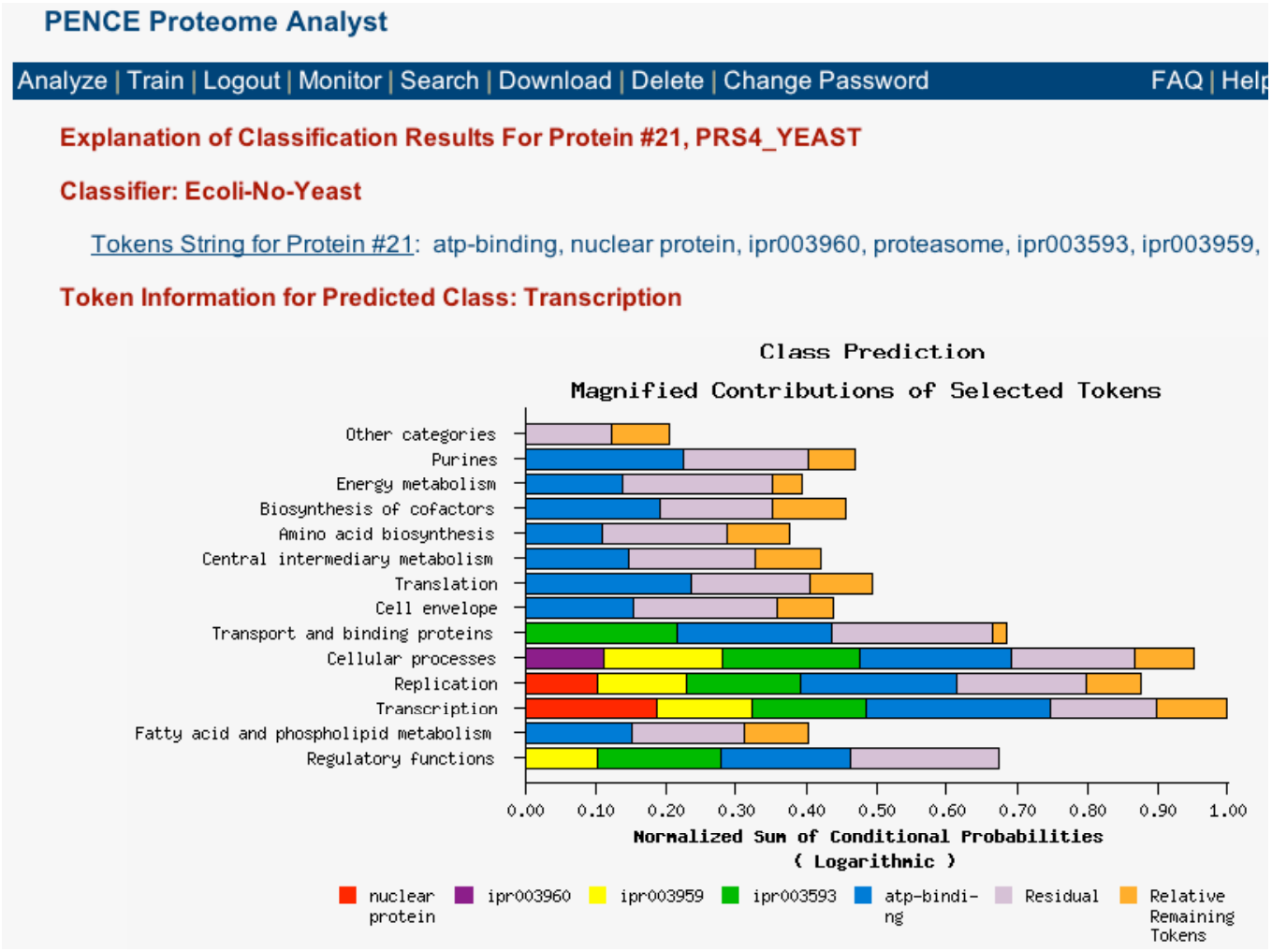


Fig. 9. Part of the *Explain* page for PRS4_Yeast.

We also computed the re-substitution accuracy (Mitchell 1997) by classifying each protein using the classifier we built and obtained a 90.0% accuracy (2832/3146). Of the 314 mis-classified sequences, 113 failed because PSI-BLAST produced no homologues with usable features. The other 201 sequences predicted different sub-cellular localizations than the training data. There are a variety of reasons for these errors. Fortunately, as shown in Fig. 5, PA identifies those sequences with the most chance of being mislabeled so they can be studied. As one example, the sequence GSHR_Human was labeled as Cytoplasm in the training data and our classifier predicted Mitochondria. In fact this protein appears both in the Cytoplasm and Mitochondria; SWISS-PROT lists both locations.

DISCUSSION

It is not only necessary for a protein prediction tool to be accurate, but it is also necessary that it can explain its predictions to the user in a clear and logical manner. This is important for two main reasons. First, it helps biologists to develop confidence in the tool and second, it can help us find and correct errors that occur in prediction.

Explaining a Prediction/Classification

Proteome Analyst (PA) provides an explanation mechanism to help users understand why a classifier made a particular classification. It allows a user to examine the classified protein itself, as well as the proteins on which the classifier was trained. The user can then examine which particular features added the most evidence to a classification. We will use the protein PRS4_Yeast as an example. If the user clicks the *Explain* hyperlink of the PRS4_Yeast protein of Fig. 2, or the *Explain* hyperlink from the output graph of Fig. 3, an *Explain* page (Fig. 9) is displayed.

Each line in the graph represents a class in the ontology and five sub-bars of a particular color represent the presence of 5 features in the training sequences. In fact, a sub-bar may represents the absence of a feature, as described in the next sub-section. However, for simplicity, in this sub-section, we will assume that sub-bars mark the presence of a feature (and

this is the case in Fig. 9) where the features are: *nuclear protein*, *ipr003960*, *ipr003959*, *ipr003593* and *atp-binding*.

Each composed bar on a single line represents the combined probability that the protein is in the class represented by the line. The red *nuclear protein* sub-bars occur in the class lines of *Transcription* and *Replication* and in no other class lines. This indicates that this feature only occurred in the training data of these two classes. The relative lengths of the sub-bars indicate the relative number of times the feature occurred in the different training sets. However, the relationship is exponential, not linear (as described in the next sub-section).

Similarly, the yellow sub-bar represents the occurrence of the feature ipr03593, which only appeared in training data for the classes: *Cellular processes*, *Replication*, *Transcription* and *Regulatory functions* and it appeared the most times (by a small margin) in Cellular Processes. It is interesting that the strongest piece of counter-evidence (against *Transcription*) is the existence of the feature *ipr003960* (purple bar), which in the training data, only occurred for sequences in the *Cellular Processes* class.

Explaining A Predictor/Classifier

PA allows the user to inspect all of the training data that contained a feature by clicking a hyperlink. In effect, this is not explaining a particular prediction – it is explaining how the predictor works. For example, PA displays all of the training sequences that contain the feature, *ipr003960*, sorted by class, as shown in Fig. 10.

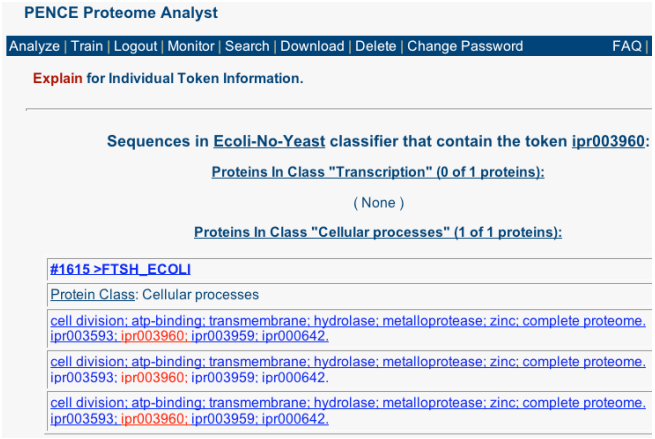


Fig. 10. Second level of *Explain* – viewing the feature *ipr003960* in the training set.

From this feature (token) information page, the user can obtain the SWISS-PROT entry of the training sequence and try to determine if it was mislabeled in the training set or if there is a legitimate divergence of function between the query protein (PRS4_Yeast) and the homologue in the training sequence (FTSH_ECOLI). In this case, since the feature is an Interpro number, the Interpro site (http://www.ebi.ac.uk/interpro) can be accessed for more information. In fact, *ipr3960* is a sub-family of *ipr3959* and represents a large family of ATPases whose key feature is a shared conserved region of about 220 amino acids that contain an ATP-binding site. The PA *Explain* mechanism lets the user browse the evidence for any prediction, right back to the training data for the predictor being used.

A More Detailed Examination of the Explain Graph

How are the lengths of the bars determined? From the NB formula, we can see that the probability, P_i, that a sequence is in class, C_i, is multiplicative on the counts $^{+}c_{ij}$, of training sequences that contain various features F_j that are in a query protein and counts $\bar{c}_{ij}$ of training sequences that do not contain features that are not in the query protein. We must map the counts to the lengths of bars in a meaningful way.

Unfortunately, it is difficult to represent multiplicative factors graphically in one dimension. It is also difficult for the human mind to intuitively grasp multiplicative factors. For example, how do we represent these product graphically and which of these products is larger and by how much - 66□150□2108□1642□1977 or 194□174□540□2215□ 2253?

We solve this problem by using logarithms. If we take the logarithm of both sides of the NB formula given previously and re-factor some terms, we obtain the formula:

$$\log(P_i) = \log(\frac{n\square^{+}c_{i1}}{n_i}) + \log(\frac{n\square^{\square}c_{i2}}{n_i}) + \ldots + \log(\frac{n\square^{+}c_{im}}{n_i}) +$$

$$\log(n_i)\square\log(\frac{K}{n^{m\square 1}})$$

Five of the colored sub-bars in Fig. 9 represent individual features and each has length proportional to one of the logarithmic terms in this formula. Since the logarithms add, it is easy to see how the features combine additively. The five particular features shown in the graph are the features that gave the greatest differential between the chosen class (*Transcription*) and the other classes. In a beta-version of PA, the user can click on any one of these five features and replace it by any other feature. The orange bar (*Relative Remaining Tokens*) represents the sum of the logarithmic terms of all features except the five selected ones. The light purple bar (*Residual*) represents the $log(n_i)$ term and accounts for the fact that there are different numbers of training sequences for each class. The last term is ignored since it is just a normalization constant that is the same size for each line of the graph. In fact, the size of all orange bars has been reduced by the same amount, in order to zoom in on the important five features for this query sequence, PRS4_Yeast. That is why the bottom line (*Regulatory Functions*) does not have an orange bar. Originally, it had the smallest orange bar but this amount was subtracted from all lines, leaving it with no orange bar and the graph was then scaled so that the longest bar has length 1.0.

Token Weights:
("Present" or "Absent" refers to the presence or absence of the token in the query sequence)

	ipr000719 (Present)	atp-binding (Present)	transferase (Present)	transmembrane (Absent)	hydrolase (Absent)	Residual	Remaining Tokens	Probability
Other categories	1.000	1.000	1.000	1,552.448	2,369.000	29.000	2,834,880.448	0.000
Purines	1.000	463.049	1,829.943	2,214.984	2,234.236	123.000	2,836,159.049	0.575
Energy metabolism	1.000	43.667	100.556	2,041.889	2,290.778	333.000	2,832,440.111	0.001
Biosynthesis of cofactors	1.000	190.440	727.187	2,305.853	2,211.133	75.000	2,835,633.853	0.052
Amino acid biosynthesis	1.000	19.944	550.376	2,350.056	2,293.224	125.000	2,834,901.352	0.004
Central intermediary metabolism	1.000	56.500	704.000	2,202.500	963.000	128.000	2,836,844.000	0.018

Fig. 11. Part of the feature (token) weight table on the *Explain* page for KAB7_Yeast.

Since the graph is a logarithmic one, small differences in the sizes of the bars actually translate to large differences in the final probabilities. For example, *Transcription* has the longest bar of length 1.0, which maps to a probability of 76.7%. *Cellular processes* has the second longest bar with length 0.96 that maps to a probability of only 20.6% and *Replication* has the third longest bar with length 0.88 that maps only to a probability of 2.7%.

Also on the *Explain* page, below the graph, is a table that contains the (n_{ij}^{+}/n_i) and (n_{ij}^{-}/n_i) and (n_i) values from the formula that were used to construct the *Explain* graph. Part of this table for the *Explain* page of protein KAB7_Yeast is shown in Fig. 11. Each row in the table corresponds to one possible class that the classifier can choose. There is one column for each of the five principle features. Each entry contains the value (n_{ij}^{+}/n_i) if the feature is present in the query sequence and (n_{ij}^{-}/n_i) if the feature is absent. This is marked under the feature name. The next column contains the residuals (n_i), which are the number of training sequences that were labeled by each class label. The next column is the sum of the (n_{ij}^{+}/n_i) for all of the other features that were contained in the query sequence plus the sum of the (n_{ij}^{-}/n_i) for the features that were absent. The last column in the table is the probability of the protein belonging to each class. Note that the logarithmic formula does not work if any ($n_{ij}^{\pm}/n_i$) is zero, since the logarithm of 0 is - . Such a term is actually compensated by the normalization (last) term that becomes + , but the formula becomes indeterminate. To compensate for this, we add 1 to each of the (n_{ij}^{+}/n_i) and (n_{ij}^{-}/n_i) terms. This is similar to the so-called aLaplacian correction and is standard practice in ML (Mitchell 1997).

The Importance of Transparency

The PA *Explain* mechanism is the most obvious example of prediction transparency in PA. The first level can be used to understand how a particular protein prediction was made. The second level can be used to understand the internal structure of a predictor – how its training data affects its predictions. Prediction transparency is very important for two reasons. First, it is hard to accept predictions unless you understand how they were made. After using the *Explain* mechanism, you gain confidence that the predictor is working properly. Second, even the best predictors will make wrong predictions! They should not be trusted blindly.

There are three important situations in which classification-based predictors fail. First, a classifier is only as good as its training data and the current databases that are used to obtain training data are far from perfect. This is why PA clearly labels "suspicious" training data as probably mislabeled, after it constructs any new classifier. This is another example of transparency in PA. PA may be directed to use these suspicious sequences in a classifier, but it clearly identifies them. A more conservative user can retrain a new classifier, without these suspicious sequences. We have found many suspicious sequences while training classifiers.

The second way that predictors can fail is if there is not enough training data to uniquely identify a single prediction class. In PA, this is characterized by a full-classifier graph (*e.g.* Fig. 3) where there are multiple bars with significant probabilities. The third way is to pick an inferior classifier algorithm, which cannot adequately use the training data to differentiate between query sequences. The difference in accuracies of the four classifiers in Table 1 is an illustration

of this since the same training and query sequences were used for all four. A trend in ML in general, and recently in bioinformatics, has been to always select the algorithm with the best accuracy. If we had followed that advice we would be using an ANN or SVM classifier in PA. But we are not! ANN and SVM do not produce classifiers with good transparency. In the production version of PA, we have opted for a classifier with up to 10% worse accuracy so that we can provide transparent predictions. We believe that this is essential in this domain, where even the best predictors make errors due to bad training data or not enough training data and these errors must be found. Would you prefer a predictor that predicts sub-cellular localization correctly 87% of the time with clear explanations or one with prediction accuracy of 92% whose predictions cannot be explained?

Why did our custom classifier attain such high-accuracy on the whole sub-cellular localization dataset, when we used the SWISS-PROTT database for features and so did Nair and Rost (2002)? The PA *Explain* feature made it easy to find the answer. Thirteen of the fourteen most important features that the PA-built classifier used were INTERPRO numbers, not KEYWORD entries.

We have constructed Proteome Analyst (PA), a web-based tool for high-throughput prediction of protein features. We support the construction of custom classification-based predictors with no programming knowledge required. Every prediction made by PA can be explained in a transparent way.

ACKNOWLEDGEMENTS

We would like to thank Cynthia Luk, Samer Nassar and Kevin McKee for their contributions to the original prototype of PA during the summer of 2001. We would also like to thank molecular biologists, Warren Gallin and Kathy Magor for their valuable feedback in using Proteome Analyst and Nair and Rost for providing us with their sub-cellular localization training data. This research was partially funded by research or equipment grants from the Protein Engineering Network of Centres of Excellence (PENCE), the National Science and Engineering Research Council (NSERC), Sun Microsystems and the Alberta Ingenuity Centre for Machine Learning (AICML).

REFERENCES

Altschul, S. F., Madden, T.L., Schäffer, A.A., Zhang,, J., Zhang, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389-3402.

Andrade M.A., Brown N.P., Leroy C., Hoersch S., de Daruvar A., Reich C., Franchini A., Tamames J., Valencia A., Ouzounis C., Sander C. (1999) Automated genome sequence analysis and annotation. *Bioinformatics*, **15**, 391-412.

Apweiler, R., Attwood, T.K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet, F., Croning, M.D.R., Durbin, R., Falquet, L., Fleischmann, W., Gouzy, J., Hermjakob, H., Hulo, N., Jonassen, I., Kahn, D., Kanapin, A., Karavidopoulou, Y., Lopez, R., Marx, B., Mulder, N.J., Oinn, T.M., Pagni, M., Servant, F., Sigrist, C.J.A., Zdobnov, E.M. (2001) The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Research*, **29**, 37-40.

Ashburner M., Ball C.A., Blake J.A., Botstein D., Butler H., Cherry J.M., Davis A.P., Dolinski K., Dwight S.S., Eppig J.T., Harris M.A., Hill D.P., Issel-Tarver L., Kasarskis A., Lewis S., Matese J.C., Richardson J.E., Ringwald M., Rubin G.M., Sherlock G. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*., **25**, 25-29.

Bairoch A. (1993) The ENZYME data bank. *Nucleic Acids Res*., **21**, 3155-3156.

Bairoch A., Apweiler R. (1997) The SWISS-PROT protein sequence database: its relevance to human molecular medical research. *J. Mol. Med*., **75**, 312-316.

Blattner F.R., Plunkett G. 3rd, Bloch C.A., Perna N.T., Burland V., Riley M., Collado-Vides J., Glasner J.D., Rode C.K., Mayhew G.F., Gregor J., Davis N.W., Kirkpatrick H.A., Goeden M.A., Rose D.J., Mau B., Shao Y. (1997) The complete genome sequence of Escherichia coli K-12. *Science*, **277**, 1453-1474.

Burges, C.J.C. (1998) A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**, 121-167.

Falquet L., Pagni M., Bucher P., Hulo N., Sigrist C.J, Hofmann K., Bairoch A. (2002) The PROSITE database, its status in 2002. *Nucleic Acids Res*., **30**, 235-238.

Friedman, N., Geiger, D., Goldszmidt, M. (1997) Bayesian network classifiers. *Machine Learning Journal*, **29**, 131-163.

Frishman D., Albermann K., Hani J., Heumann K., Metanomski A., Zollner A., Mewes H. W. (2001) Functional and structural genomics using PEDANT. *Bioinformatics*, **17**, 44-57.

Gaasterland T., Sensen C.W. (1996) MAGPIE: automated genome interpretation. *Trends Genet*., **12**, 76-78.

Gallin, W.J. and Spencer, A.N. (2001). Evolution of Potassium Channels. In Archer, S.L. and Spencer A.N. (eds), *Potassium Channels in Cardiovascular Biology*, Kluwer, Dordrecht, 3-16.

Harris N.L. (1997) Genotator: a workbench for sequence annotation. *Genome Res*., **7**, 754-762.

Hubbard T., Barker D., Birney E., Cameron G., Chen Y., Clark L., Cox T., Cuff J., Curwen V., Down T., Durbin R., Eyras E., Gilbert J., Hammond M., Huminiecki L., Kasprzyk A., Lehvaslaiho H., Lijnzaad P., Melsopp C., Mongin E., Pettett R., Pocock M., Potter S., Rust A., Schmidt E., Searle S., Slater G., Smith J., Spooner W., Stabenau A., Stalker J., Stupka E., Ureta-Vidal A., Vastrik I., Clamp M. (2002) The Ensembl genome database project. *Nucleic Acids Res*., **30**, 38-41.

Jones, D.T. (1999) Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol*., **292**, 195-202.

Kitson, D.H., Badretdinov, A., Zhu, Z.Y., Velikanov, M., Edwards, D.J., Olszewski, K., Szalma, S., and Yan, L. (2002) Functional annotation of proteomic sequences based on consensus of sequence and structural analysis. *Brief Bioinformatics*, **3**, 32-44.

Kohavi, R. and John, G.H. (1997) Wrappers for feature subset selection. *Artificial Intelligence*, **97**, 273-324.

Mitchell, T.M. (1997) *Machine Learning*. McGraw-Hill, N.Y.

Nari, R. and Rost, B. (2002) Inferring sub-cellular localization through automated lexical analysis. *Bioinformatics*, 18, S78-S86.

Overton GC, Bailey C, Crabtree J, Gibson M, Fischer S, Schug J. (1998) The GAIA software framework for genome annotation. *Pac Symp Biocomput*., 291-302.

Raychaudhuri S., Chang J.T., Sutphin P.D., Altman R.B. (2002) Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome Res*., **12**, 203-214.

Riley M. (1993) Functions of the gene products of Escherichia coli. *Microbiol Rev*., **57**, 862-952.

Sahami, M. (1999). Using Machine Learning to Improve Information Access. Ph.d. thesis, Computer Science Department, Stanford University.

Schug J., Diskin S., Mazzarelli J., Brunk B.P., Stoeckert C.J. Jr. (2002) Predicting gene ontology functions from ProDom and CDD protein domains. *Genome Res*., **12**, 648-655.

Xie H., Wasserman A., Levine Z., Novik A., Grebinskiy V., Shoshan A., Mintz L. (2002) Large-scale protein annotation through gene ontology. *Genome Res*., **12**, 785-694.