SPINE: An Integrated Tracking Database and Data Mining Approach for Identifying Feasible Targets in High-throughput Structural Proteomics

Paul Bertone[1,2], Yuval Kluger[2,3], Ning Lan[3], Deyou Zheng[4], Dinesh Christendat[5], Adelinda Yee[5], Aled M. Edwards[5], Cheryl H. Arrowsmith[5], Gaetano T. Montelione[4], and Mark Gerstein[3*]

[1]Department of Molecular, Cellular, and Developmental Biology, Yale University, New Haven, Connecticut 06520, USA; [2]These authors contributed equally to this work; [3]Department of Molecular Biophysics and Biochemistry, Yale University; [4]Center for Advanced Biotechnology and Medicine and Department of Molecular Biology and Biochemistry, Rutgers University, Piscataway, New Jersey 08855, USA; [5]Ontario Cancer Institute and Department of Medical Biophysics, University of Toronto, Ontario, Canada M5G 2M9.

* Corresponding author
Telephone: (203) 432-6105, (203) 432-5605
Fax: (360) 838-7861
E-mail: Mark.Gerstein@yale.edu

Author contributions
PB, NL, DZ, GM, MG: Database design
PB, NL: Database implementation
DC, AY, AL, CA, GM: Experimental results
YK, PB, MG: Data mining
MG: Project leader

Abstract

High-throughput structural proteomics is expected to generate considerable amounts of data on the progress of structure determination for many proteins. For each protein this includes information about cloning, expression, purification, biophysical characterization, and structure determination via NMR spectroscopy or X-ray crystallography. It will be essential to develop specifications and ontologies for standardizing this information to make it amenable to retrospective analysis. To this end, we created the SPINE database and analysis system for the Northeast Structural Genomics Consortium. SPINE, which is available at bioinfo.mbb.yale.edu/nesg or nesg.org, is specifically designed for enabling distributed scientific collaboration via the Internet. It was designed not just as an information repository but as an active vehicle to standardize proteomics data in a form that would enable systematic data mining. The system features an intuitive user interface for interactive retrieval and modification of expression construct data, query forms designed to track global project progress, and external links to many other resources. Currently, the database contains experimental data on 985 constructs, of which 740 are drawn from *M. thermoautotrophicum*, 123 from *S. cerevisiae*, 93 from *C. elegans*, and the remainder from other organisms. We developed a comprehensive set of data mining features for each protein, including several related to experimental progress (e.g. expression level, solubility, and crystallization) and 42 based on the underlying protein sequence (e.g. amino-acid composition, secondary structure, and occurrence of low-complexity regions). We demonstrate in detail the application of a particular machine learning approach, decision trees, to the tasks of predicting a protein's solubility and propensity to crystallize based on sequence features. We are able extract a number of key rules from our trees, in particular that soluble proteins tend to have significantly more acidic residues and fewer hydrophobic stretches than insoluble ones. One of the characteristics of proteomics data sets, currently and in the foreseeable future, is their intermediate size (~500 to ~5000 data points). This creates a number of issues in relation to error estimation. Initially, we estimate the overall error in our trees based on standard cross-validation. However, this leaves out a significant fraction of the data in model construction and does not give error estimates on individual rules. Therefore, we present alternate methods to estimate the error in particular rules.

Introduction

The role of computational techniques in biological research is certain to increase with the advent of genomics. Databases in particular have become invaluable tools in molecular biology. The current landscape of biological databases includes large, general purpose repositories for nucleotide sequences such as GenBank [1], DDBJ [2], and EMBL [3] and protein sequences like PIR [4], SWISS-PROT [5], and the Protein Data Bank [6]. Likewise, there are many specialized databases storing information related to model organisms such as SGD [7], MIPS [8], and FlyBase [9], comparative genomics [10,11], gene expression [12,13], protein-protein interactions [14,15], and protein motions [16]. The PartsList [17] system encapsulates the results of surveying the occurrence of folds and protein features in genomes [18,19], while the Presage database [20] was recently developed for target selection in structural genomics. Software tools are also available for the creation of project-specific laboratory information management systems (LIMS), such as LabBase [21].

Many biological databases are developed and maintained strictly for warehousing purposes, without consideration of analyses that may be performed on the data. Conversely, computational studies are often performed outside of the context of information management, without a clear connection to biological reality. Our work explores a fusion of these two processes, where database design is influenced by analytical requirements.

Such an undertaking requires a centralized repository to integrate and manage the data generated, coupled with strategies for subsequent computational analysis. By maintaining a shared infrastructure accessible to all the participating members of a project, distributed access to large subsets of data is possible. This not only promotes collaborative effort among investigators by providing a common information exchange platform, but also avoids costly and time-consuming duplication of experimental work. Further, data is maintained in a consistent format across many laboratories and investigators, promoting further analysis.

To this end we have developed the SPINE database and analysis system, an integrated approach to interactive database system design and computational analysis in a distributed framework, using the recently formed Northeast Structural Genomic Consortium (www.nesg.org) as a model for multi-laboratory collaborative research. The system is designed to generate standardized data files from user-definable subsets of the proteomics information entered into the database, which are then used for classification tasks. Key issues in effective data mining are introduced, with emphasis on decision trees, a supervised machine learning approach. We conclude with a discussion of prediction results for several macromolecular properties based on features derived from the database contents.

Database System Requirements

SPINE (structural proteomics in the Northeast) was designed for the Northeast Structural Genomics Consortium (NESG), a multi-institutional collaboration for the high-throughput determination of protein structures on a genomic scale, with an emphasis on model eukaryotes. The project coordinates the identification of suitable target proteins and the production of expression constructs from which proteins will be purified, followed by biophysical characterization via circular dichroism, and a series of NMR or X-ray crystallography studies to determine tertiary structures. Experimental data generated by this project was used for the development of a distributed data archival and analysis framework suitable for laboratory information management, standardization of experimental parameters, and data mining techniques. Several views of the database developed for this project are shown in Figure 1.

A critical issue in designing a system of this kind is determining the fundamental "unit" to be tracked by the database. In many cases this process is straightforward. For example, a database suited to classical genetics would most likely record parameters related to the expression of individual genes under various experimental conditions and the function of

their associated proteins. In this case, the fundamental unit of information would most likely be the gene. However, in our case the solution is not so obvious, as a variety of choices exist. A database record can be based on any number of entities, such as the protein itself, a construct used to express the protein, a specific experimental preparation, or a more abstract "target" representation encompassing multiple proteins in a particular family. The most appropriate representation depends on the scope of the project and the relative stability of the data types under consideration.

An obvious candidate for the fundamental database unit is the protein. However, in certain instances homologous proteins from other organisms prove more experimentally tractable than the actual target; this scenario would be a source of confusion when maintaining a resource based on proteins. An alternative is to focus on the expression construct made for a given protein. Multiple constructs can be made for a single protein, because a construct could be designed to express only a single domain from a complex protein, or contain a slightly altered protein sequence that aids in protein production and structure determination. This one-to-many relationship between target proteins and their associated expression constructs would imply that several database entries might be related to the same target. A third option is to use the specific preparation associated with each experiment, where a database record could represent a set of conditions by which a protein sample is prepared. An immediate concern with this representation is that protein preparations will vary constantly, requiring an unforeseeable number of relational tables to accommodate their parameters.

Because multiple constructs can be generated for each target, the single protein representation is too limited for our purposes. Conversely, experimental conditions for individual protein preparations are highly variable, and it was decided that this data should be compiled separately. From these candidates, it was decided that the expression construct captured the most appropriate level of detail for this project. It was selected as the basic unit to be tracked by the database, essentially recording the best experimental results for the expression, purification, and characterization of each target protein.

To address these requirements, software components were developed for entry and updating of expression construct records, database searching, bulk data retrieval, and tracking the global progress of the entire project. Intuitive HTML form-based interfaces were implemented to facilitate distributed Internet access from participating laboratories. The implementation of the database system is described in detail in Figure 2. The modular organization of software components permits relatively straightforward implementation of additional functionality. This aspect is independent from the underlying database architecture, allowing a great deal of programming flexibility while maintaining strict compliance to the standardized data types established for various experimental parameters.

Database Fields

The SPINE database fields were compiled with subsequent computational analysis in mind. Information having disparate formats and types would make data mining impossible, so an important role of the system is the standardization of expression construct data sets. Using a centralized data repository having a defined table structure, information is maintained in a consistent format regardless of the investigator or laboratory where the data originates. Another benefit is the introduction of numerical values in place of the text descriptors sometimes used by experimentalists.

To accommodate the needs of various Consortium projects where different experimental methodologies are used, principal investigators from several laboratories were involved in the process of selecting the most appropriate information to be tracked by the system. Fields from existing data sets were used to develop a consensus of experimental parameters, and this was adapted to the current database framework.

A listing of the fields used for the prototype database is shown in Table 1. The information maintained by the system initially comprised 63 fields for protein sequences, cloning parameters, expression level and purification yield, and data derived from biophysical characterization and structural biology experiments (oligomerization, CD, HSQC, NMR, and X-ray crystallography). In addition, a number of fields are devoted to

keeping track of the laboratory and investigator responsible for working with the target protein, dates when experiments were performed, comments relating to experimental conditions for each group of related fields, and variable access levels for individual database records. The database is not intended to manage all aspects of experimental research; rather it is designed to standardize and track key parameters related to structural proteomics. However, the system does include user accounts, transaction history information, and some lab management tables.

The development of this system is an ongoing project. Following the establishment of a prototype database, additional features were implemented to reflect the needs of the Consortium laboratories, and its schema was expanded over a number of relational tables (Figure 3). The current design allows various groups of database fields to be accessed and updated, depending on the type of experiments typically performed by different investigators. Users may limit the parameters to be input for a database record to a subset of fields that are relevant to their work by selecting forms specialized for NMR spectroscopy, X-ray crystallography, etc. By using only those fields that are applicable to a particular experimental process, navigating through long forms with potentially unused elements is avoided.

Users of the database access the system through a password-protected interface. The instantiation of individual user spaces aids in managing proprietary data associated with each experimentalist. This modification is beneficial in terms of designing more transparent user interfaces. For example, investigator profiles are maintained which keep track of routinely used field values and experimental methods, allowing the system to complete certain fields automatically.

For many experimental methods, a data file is generated comprising an entire set of results distinct from the information tracked by the main database. The inclusion of these parameters into the existing infrastructure would be beyond the scope of the system; HSQC spectra, X-ray diffraction data, and NMR assignments can span large files that would be impractical to incorporate directly into database tables. Instead, these are stored

on a separate file server, and the associated URL addresses are recorded in construct records and linked to each record display. Thus, a key feature of the system is maintaining a central collection of pointers to additional experimental data sets. This mechanism is, of course, extended to allow pointers into other information repositories associated with the project, for instance, into a crystallization database or a list of targets. We also link the system with other protein sequence and structure resources, such as SWISS-PROT [5], ProtoMap [22], GeneCensus [17,18,19], PartsList [23], SCOP [24], and CATH [25].

User Interaction and Dynamic Content Modification

The design of the system's front end allows expression construct records to be entered, edited, and retrieved by individual users without frequent intervention of a database curator. An important goal in this work is to design a system that works in a practical laboratory setting. That is, the software is operationally robust and straightforward, so that using it on a regular basis will not disrupt workflow. The system provides a consistent and intuitive user interface to complex database functions, as well as error recovery features when conflicting or incomplete information is submitted. Search functions were developed for the intelligent retrieval and display of information from the database, as well as the ability to generate bulk dumps of large subsets of data records and protein sequences in interchangeable file formats, including CSV and XML.

As experimental work progresses on a given target, additional data is collected which may have been unavailable at the time its expression construct record was created. Therefore, an essential requirement of the database is the ability to recall records to alter or augment their associated information. Consequently, the contents of individual database records are changing over time in a user-mediated fashion, in contrast to more archive-oriented resources. This imposes additional sets of operational considerations, requiring provisions to ensure internal ID consistency and overwrite protection when users enter or modify database records.

System Functionality

Data Entry and Editing

The creation of expression construct records is generally performed on a per-instance basis, using a series of HTML forms. Database records are keyed on an identifier string that can be selected by the investigator or generated automatically by the system. The custom identifier feature is particularly useful in cases where a construct for a given protein is derived from an organism different from the target organism in which the protein originates. For example, the identifier HTEC5 could be used to represent the fifth *E. coli* expression construct (EC) for a human target protein (H), originating from a Toronto laboratory (T). The data entry procedure is designed to be simple and intuitive. During the process of generating identifiers and creating new records, key parameters are retained as subsequent web forms are encountered, to minimize effort and eliminate user error. This process is depicted in Figure 4.

As new experimental data is accumulated, existing records must be augmented and modified. This is accomplished via editor forms identical in layout to the data entry forms. Database attribute values are recalled and inserted into their corresponding editor fields, where they may be modified. After changes have been made and any additional data have been entered, the record is updated to reflect the new information.

Searching the Database and Visualizing Progress

The retrieval of records from the database is accomplished through the use of a search engine interface (Figure 5A), where a variety of terms may be selected and combined with Boolean connectives. Based on the values of the elements submitted via the interface form, the software builds an SQL query to execute against the database and returns any records matching the search terms (Figure 5B). The subset of database records returned by the search may be optionally downloaded as a CSV-formatted text file, suitable for importing into another database or spreadsheet program. Individual expression construct records are displayed in a static web page (Figure 5C), with database fields organized in a logical hierarchy. A number of local and distributed Internet

resources are automatically linked to record display pages, such as Protein Data Bank searching, organism-specific databases, and specialized structural annotation reports.

To provide a global view of project growth, programs were developed to summarize the nature of the database holdings, and illustrate the relative progress made on target proteins (Figure 1A). Using a subset of the main search engine functionality, users can recall sets of database entries and display them in a large table, organized to represent a timeline in the structure determination process. Advanced features allow users to reconfigure the display to generate a custom table that presents any combination of database fields in lieu of the standard table layout.

Data Mining Applications for High-throughput Proteomics

The success of the high-throughput aspect of structural proteomics relies on the optimization of target selection and experimental protocols. This, in turn, involves identifying proteins that can be readily expressed, solubilized, purified, and crystallized under a given set of standard conditions (i.e. the most tractable instances). These factors will strongly influence whether or not a given protein is pursued for X-ray or NMR structure determination. One of the main goals of the SPINE system was to capture the data in a way that made it suitable for subsequent analysis. In the following sections, we present a representative application: classification of soluble proteins using decision trees. Before presenting the details, it is worthwhile to review some key elements of this approach.

Machine Learning Concepts

The term machine learning applies to a wide range of computational methodologies. However, the models most suitable for our applications belong to the class of algorithms that employ supervised learning. Under supervised learning, the classification process consists of two phases: training and testing. The set of all available examples or instances (formally termed input vectors) is divided into two nonintersecting sets. The first set is

used to train the model. During this phase, correct classifications of the examples are known *a priori*. Supervised learning strategies rely on this information to adjust the performance of the model until the classification error rate is sufficiently reduced. Learning is no longer performed after training is completed; instead, unseen instances in the test set are classified according to the partitioning established during the training phase. The performance of a learning algorithm is determined by its ability to correctly classify new instances not present in the initial training set.

The features of each sample can be represented as a vector that corresponds to a point in an *n*-dimensional space. Classification is then performed by partitioning this feature space into regions, where most of the points in a region correspond to a particular category. The goal in training classifiers is to find an optimal partitioning of the input space separating the highest number of disparate examples. An ideal classifier will demonstrate strong predictive power, while explaining the relationships between the variable to be predicted and the variables comprising the feature space.

Machine learning Applications to Proteomics Data

One property of a proteomics feature set that one must adhere to is the appropriate time frame in which classifications are performed. In many cases the experimental results are serially related, constraining the composition of useful training sets to expression constructs having *a priori* prediction data. For example, one cannot expect to optimally classify crystallization targets if the available training set contains experimental results only up to the expression stage, because the available feature set will not contain a response variable for crystallization. Conversely, it is entirely possible to classify proteins based on some property corresponding to an earlier experimental stage - e.g., solubility data has already been gathered for proteins having HSQC spectra, enabling one to train a classifier to partition these proteins based on solubility information.

While there are many possible issues that data mining can address in relation to the proteomics data collected by the Consortium, we have focused on protein solubility prediction due to the importance of this property and the availability of a large set of

Methanobacterium thermoautotrophicum expression construct records having solubility measurements. The size of this data set provides the best opportunity for generalization during training, increasing an algorithm's prediction success when presented with new examples. An accurate prediction method for this property can also be an extremely useful tool, as insolubility accounts for almost 60% of experimentally recalcitrant proteins [26]. Here, we refer to solubility as "soluble in the cell extract", a property that is correlated with, but not necessarily identical to, the solubility of a purified protein.

In a supervised learning approach for solubility prediction, the training set consists of a subset of input vectors extracted from the database, and is used by the classifier model to partition the sample space based on solubility, the dependent variable to be predicted. After training, the feature space will be partitioned into two regions: one containing proteins labeled as soluble, and another with proteins labeled as insoluble. The second part of this approach is to determine a trained classifier's ability to generalize to unseen examples, by presenting the model with a test set containing new feature vectors and reevaluating its performance.

M. thermoautotrophicum Data Set

A data set comprising 562 proteins from the *M. thermoautotrophicum* genome was compiled from the database and used for machine learning. Although SPINE currently holds 740 construct entries for this organism, 178 of these do not have solubility information and thus are not suitable for classification. As summarized in Table 2, a total of 42 features were extracted from the corresponding protein sequences, such as amino acid composition, hydrophobicity, occurrence of low-complexity regions, secondary structure, etc. Combined with the database fields highlighted in Table 1, these features comprise the input vector used for the classification study presented here.

To identify which proteins were used for this study, we constructed a "frozen" version of the database at bioinfo.mbb.yale.edu/nesg/frozen. This contains the entries reported here and will not change in the future. The *M. thermoautotrophicum* protein expression

constructs on which the analysis was performed are also highlighted in the frozen database.

It should be noted that prediction results for a proteomics data set may exhibit some degree of specificity to the expression vectors and experimental conditions of cell growth, induction, etc. used for protein purification. A characteristic of this specific *M. thermoautotrophicum* data is the uniform set of conditions that were used to prepare protein samples [26]. Additionally, the experimental targets selected by the Consortium consist largely of non-membrane proteins, so the available data set is biased in this regard.

Decision Tree Analysis

The selection of an appropriate learning algorithm depends on several factors, such as the type of data to be classified (numeric, symbolic), the number of available examples in the data set, and how many of the examples are likely to be noisy or inaccurate. Computational considerations such as processing time, memory limitations, and feasibility of implementation are also influential. Another issue is the degree of desired interpretability of the results, which is largely determined by the representation language used by a given algorithm. One method may exhibit advantages in interpretation, but may generalize less optimally than another (or vice versa). The most appropriate balance between prediction success and interpretation depends on which quality is more important for the application. We evaluated a number of different approaches for this study, including neural networks, decision trees, support vector machines, Bayesian networks, and linear discriminants. Here we focus on decision trees due to the relative ease of interpretability afforded by the model.

Model Description

Decision tree learning [27,28] is a widely used and effective method that can partition data that is not linearly separable (Figure 6). An individual object of unknown type may be classified by traversing the tree. At each internal node, a test is performed on the

object's value for the feature expressed at that node (often called the splitting variable, e.g. alanine composition). Based on this value, the appropriate branch is followed to the next node. This procedure continues until a leaf node is reached and the object's classification is determined. In classifying a given object, a variable number of evaluations may be performed or omitted, depending on the path taken when the tree is traversed. In this manner, a heuristic search is performed to find a compact, consistent solution that generalizes to unseen examples.

During training, the tree is grown in two stages: (1) splitting the nodes and (2) pruning the tree. A common criterion for binary node splitting entails maximizing the decrease in an impurity measure, such as residual mean deviance. The lower the deviance, the better the tree explains the variability in the data. A binary split for a continuous feature variable v is of the form $v < threshold$ versus $v > threshold$; for a "descriptive" feature, a binary split divides the feature's value range into two classes. The size of the decision tree necessary to classify a given set of examples varies according to the order in which properties are tested, and growing a tree usually has the effect of overfitting the training set. A common strategy in most pruning algorithms is to choose the smallest tree whose error rate performance is closest to the minimal error rate of the larger, original tree, as this is the model most likely to correctly classify unknown objects. Pruning is particularly important with noisy data (where the distribution of observations from the classes overlap) as growing the tree in this case will usually overfit the training set.

A number of advantages are evident in the decision tree model. Classification can be based on an arbitrary mixture of symbolic and numeric variables, and (for axis-parallel splitting) one is not required to scale the variables relative to each other. The model is generally robust when presented with missing values. In addition, straightforward and concise rules can be inferred from the tree by following the path from root to leaf nodes.

Feature selection

We used decision trees to partition the 562 *M. thermoautotrophicum* proteins into soluble and insoluble classes, based on a subset of the features listed in Table 2. The features that

are relevant to a given problem domain are often unknown *a priori*, and removing those which are redundant or irrelevant can produce a simpler model which generalizes better to unseen examples. Automated feature selection attempts to find a minimal subset of the available features, in order to either improve classification performance, or to simplify the model's structure while preserving prediction accuracy [29]. Typically, a search algorithm is used to partition the available feature set. Classifiers are then trained on the feature combinations presented by the search algorithm to identify those features which have the greatest impact on learning. In our case, we used a genetic algorithm [30] to search the space of possible feature combinations; the relevance of individual feature subsets was estimated with several machine learning methods, including decision trees and support vector machines [31]. We arrived at a feature subset consisting of the amino acids E, I, T, and Y, combined compositions of basic (KR), acidic (DE), and aromatic (FYW) residues, the acidic residues with their amides (DENQ), the presence of signal sequences and hydrophobic regions, secondary structure features, and low-complexity elements. These are highlighted in Table 2.

Decision Tree Results

The trees that were trained on this data set had a misclassification rate of 12%. Decision trees built on the training set are always overly optimistic, and contain a large number of nodes. Only the upper region of the tree is significant in terms of yielding a generalized concept, and this is the segment from which useful rules can be derived. After training and pruning of the decision trees, we extracted several classification rules for distinguishing between soluble and insoluble proteins, as described in Figure 7. Two trees are shown in this example. Figure 7A illustrates the upper 5 levels of a decision tree built on the entire set of 562 proteins and subjected to cross-validation. The tree in Figure 7B was trained on a 375-protein subset of the data and tested with the remaining 187.

Both of these exhibit simple rules for classifying soluble and insoluble proteins, illustrated by the green and red paths on either side of the root node. In the case of Figure 7A, soluble proteins are selected by the right branching path of the tree, provided their amino acid sequences have a combined aspartate and glutamate composition (represented

as *C*(DE)) of at least 18%. This path further classifies soluble proteins based on the length of their sequences, although the most discriminating variable is clearly the presence of acidic residues. Following the left branching path of the tree, insoluble proteins are selected based on the conditions that their sequences contain 1) fewer than 18% acidic residues (*C*(DE)), a long (at least 20-residue) stretch of amino acids with a minimum hydrophobicity of less than –0.78 kcal/mol on the GES scale [32] (labeled Hphobe), and a combined composition of the acidic amino acids and their polar amides (*C*(DENQ)) under 16%.

The decision tree depicted in Figure 7B further isolates the two most discriminating features: acidic residues composition and the presence of a hydrophobic stretch. Aside from their metal ion binding abilities, aspartic and glutamic acid are negatively charged due to their carboxyl side chains. These highly polar residues are often found on the surface of globular proteins, where they can interact favorably with solvent molecules. They in fact have the highest charge density per atom of all the amino acids, a property obviously associated with solubility. The hydrophobic region identified is not long enough to be considered a transmembrane helix, but clearly identifies an “adhesive” area of the protein.

Decision tree learning produces a variety of tree topologies depending on the specific data and features used for training. We divided the 562-protein data set into random training and testing sets of 66% and 33% of the input vectors, respectively, and built decision trees using all of the available features. A number of interesting patterns emerge based on the utilization of classification features in various trees. Examining the decision tree paths reveals intricate sorting based on amino acid composition in addition to the most widely used features. For example, a rule was discovered which selects soluble proteins having greater than 18% DE composition, fewer than 8% arginine and greater than 3% lysine residues. Another tree exhibited similar prediction success by combining arginine and lysine into a common splitting variable immediately following the 18% DE rule, identifying soluble proteins having less than 14% KR composition. However,

aspartic and glutamic acids were then isolated in lower levels of the tree, achieving a finer partitioning by sorting on the individual acidic residues.

Cross-validation

Overfitting can occur when a model performs well on the training set, but fails to generalize to unseen examples. In these instances, the algorithm has partitioned the data too finely and has mapped a decision surface to the training data that too closely follows intricacies in the feature space without extracting the underlying trends, essentially "memorizing" the training set. In practice, we can say that if an alternate learning solution exists with a higher error rate but generalizes better over all available input vectors, overfitting has occurred. One way of studying (and hence subsequently preventing) overfitting is cross-validation, which gives an estimate of the accuracy of a classifier based on resampling.

Stratified 10-fold cross-validation was performed on the decision trees, where each successive application of the learning procedure used a different 90% of the data set for training, and the remaining 10% for testing. Each of these training sets produced different trees from those constructed based on the entire data set. Using the testing sets for validation with their corresponding tree models, we took the sum of the number of incorrect classifications obtained from each one of the ten test subsets, and divided that sum by the total number of instances that have been used for testing (i.e., the total number of instances in the data set), thereby producing the estimated error for the entire tree. This cross-validation approach resulted in an overall prediction success of 61-65% over the various data subsets. This does not correspond directly to the decision tree performance based on the entire data set, as cross-validation results are produced from many different partitions of the training and testing sets.

While typically used for error estimation, cross-validation is not optimal for medium-sized or "mesoscale" data sets, such as our proteomics set. This is because the procedure excludes a large fraction of the data during training, resulting in insufficiently sized testing sets. Consequently, other non-cross-validated estimates of classification error

have been developed. In the next section we apply one such method, called pessimistic error estimation [27].

Rule Assessment

Regardless of the specific method of error estimation used, some paths, i.e. sequences of rules, within the decision tree may perform significantly better than others. These rules provide a straightforward way for others to apply the classification results in a practical context. Moreover, a few simple rules extracted from the tree may be considerably more robust to changes in the underlying data than the original tree topology. Consequently, we describe in this section a way to measure the quality of a particular rule - in constrast to the overall estimate of a tree's performance reported above.

For this rule assessment, we do not perform cross-validation at all due to the scarcity of the data underlying any particular rule. Instead, we use Quinlan's pessimistic estimation method, calculating a rule's accuracy over the training examples to which it applies and then calculating the standard deviation in this estimated accuracy assuming a binomial distribution. More specifically, given the set C of training cases at node Q, its majority class, and the number of cases outside that class, error-based pruning interprets C as a binomially distributed sample with well-defined confidence limits, and estimates Q's error rate as the upper limit on its posterior probability distribution. Equivalently, for a given confidence level, the lower bound estimate is then taken as the measure of the rule performance.

The default accuracy of choosing a soluble protein in our data set is defined by S/T, where S is the number of soluble proteins and T is the total number of proteins. The accuracy of the rule that predicts solubility is s/t, where s is the number of proteins reaching the leaf node at the end of a decision path, and t is the total number of proteins reaching that node. It is straightforward to evaluate the probability that a randomly chosen rule will do as well as or better than a decision rule with accuracy s/t. This probability is given by:

$$\sum_{i=s}^{\min[t,S]} \frac{\binom{S}{i}\binom{T-S}{t-i}}{\binom{T}{t}}$$

Note that the sum is over the hypergeometric distribution. Small values of this measure correspond to good rules because this means there is a small probability that a rule has arisen by chance.

For example, the branching of the tree at the root, based on the condition that the overall composition of aspartate and glutamate in protein sequences is greater than 18%, defines a rule which classifies many proteins as soluble. This rule has an observed accuracy of 108/136 (0.79) over the training set, and a probability of 6×10^{-9} of arising by chance. We must take into account the fact that the observed accuracy is overly optimistic, and correct it by subtracting 1.96 times the binomial standard deviation (for the lower bound of a 95% confidence interval). For $t > 30$, the binomial distribution can be approximated by the normal distribution.

The probability that a random variable X, with mean 0, lies within a certain confidence range of width $2z$ is $P(-z < X < z) = c$. For a normal distribution, the value of the confidence c and the corresponding values of z are given in standard tables. In our case we want to standardize s/t. To do this, we first assert that the observed success rate s/t is generated from a Bernoulli process with success rate b. If t trials are taken, the expected success of the random variable s/t is the mean of a Bernoulli process b and the standard deviation:

$$\sqrt{\frac{b(1-b)}{t}}$$

The variable s/t can be standardized by subtracting it from the mean b and dividing by the standard deviation. The standardized random variable X is defined as:

$$\frac{\frac{s}{t} - b}{\sqrt{\frac{b(1-b)}{t}}}$$

Assuming that t is large enough, the distribution of X approaches the normal distribution. As mentioned above, the probability that the random variable X with mean 0 lies within a certain confidence range of width $2z$ is $P(-z < X < z) = c$, or explicitly:

$$P\left(-z < \frac{\frac{s}{t} - b}{\sqrt{\frac{b(1-b)}{t}}} < z\right) = c$$

Choosing a confidence probability c corresponds to a particular value of z (note that standard Z values are given for the one tailed $P(X < z)$. For example, $P(X < z) = 5\%$ corresponds to $P(-z < X < z) = 90\%$). Solving for the value of b will give us the range of the success rate, and we will choose the lower bound to find the pessimistic error rate (success rate + error rate = 1; taking the largest error that corresponds to the smallest success rate will yield the pessimistic error rate).

Inspecting the argument of the above equation, we can solve for b at the boundaries $+z$ and $-z$, i.e.,

$$\frac{\frac{s}{t} - b}{\sqrt{\frac{b(1-b)}{t}}} = z \text{ , } \frac{\frac{s}{t} - b}{\sqrt{\frac{b(1-b)}{t}}} = -z$$

Then we can express the confidence range as:

$$b = \frac{\frac{s}{t} + \frac{z^2}{2t} \pm z\sqrt{\frac{s}{t^2} - \frac{p^2}{t^3} + \frac{z^2}{4t^2}}}{1 + \frac{z^2}{t}}$$

and take the lower limit. Taking the pessimistic lower bound estimate for a 95% confidence interval gives an overall 0.71 success ratio, in contrast with the default rule at the root of the tree, which has a success rate of 330/562 (0.59). The probability of this rule occurring by chance is less than 0.1%.

A statistically valid approach to estimate the true error (e_t) of a hypothesis within a 95% confidence interval is given in terms of a sample error (e_s) and the sample size n ($n > 30$):

$$e_t = e_s \pm \sqrt{\frac{e_s(1-e_s)}{n}}$$

In addition to cross-validation and error estimation, model combination techniques were applied using decision trees derived from random subsets of the available data. These methods included bagging (bootstrap aggregating) and boosting [33], where each new model is influenced by the performance of those built previously and is trained to classify instances handled incorrectly by earlier ones. No significant improvement in prediction rates was found with any of these approaches. Similarly, the approach of stacking several different classifiers, such as a decision tree with a support vector machine, to another higher-level meta-learner (e.g., another decision tree classifier) also did not change the prediction accuracy.

Identification of potential crystallization targets

We also performed machine-learning analyses on other aspects of the proteomics data set, such as the potential for crystallization. An example decision tree built to classify 64 proteins based on their tendency to crystallize is shown in Figure 6. From this result, it appears that the top-level rule in the tree, aspartate composition of greater or less than 4.5% is a discriminating feature. Significantly less data is available for this classification

task than for solubility prediction; hence these preliminary results are not statistically robust. When more data becomes available, we should be able to derive rules relating other protein attributes using the decision tree approach.

Discussion

Comprehensive data management practices coupled with computational analysis can be a powerful tool for large-scale proteomics. An interactive, dynamically modifiable database is an important component in collaborative research, enabling global protein target prioritization and synchronization of efforts across many laboratories. If data resources are designed for subsequent analysis, data mining strategies can be an effective way to make sense of experimental data and discover hidden trends. Implementing robust, standardized archival procedures to maintain data from disparate sources is critical to the success of large-scale projects where many laboratories may be collaborating. In turn, the effective application of retrospective (post-experimental) analysis methods is dependent upon the availability of comprehensive data sets having standard formats easily parsed by computer programs.

By its nature, large-scale genomics and proteomics research cannot be performed by a conventional single-investigator laboratory. It will be carried out in large central facilities or via consortium of many laboratories. Our approach is designed to facilitate the latter research model. This approach enables not only the integration of data from various sources, but also the formulation of statistical predictions of various macromolecular properties, which can potentially enhance the efficiency of laboratory research.

In particular, decision tree models feature a number of practical advantages, such as the straightforward interpretation of results, ability to mix numeric and symbolic features, and invariance to numeric scaling. The ability to devise prediction rules from the paths through the tree is perhaps the most powerful feature of this approach. Eventually, we plan to do a comparative study of several machine learning algorithms, to assess the

capabilities of various methods for predicting macromolecular properties of new proteins based on the training sets produced by the database.

Database Extensions: Sparse Data Records and Multiple Expression Constructs

The prototype database system is currently implemented as a multi-table relational model. The limited scalability of this design may become problematic as the system expands to capture more diverse experimental data, resulting in a larger number of unused fields. To circumvent this "sparse matrix" problem, future versions of the system are moving toward the entity attribute value (EAV) representation [34]. This design would allow various sets of database fields to be accessed and updated, depending on the type of experiments typically performed by different investigators. Efforts are ongoing to standardize and incorporate more experimental data into this format, so that computational methods can be applied to a wider range of features.

A related issue concerns the way multiple expression constructs having a shared protein target should be considered for analysis. In order to predict various properties of proteins, it may be necessary in some cases to collapse the data from related expression constructs to the target protein level. Although this problem was not encountered with the data sets used for the studies presented here, it remains to be seen which approaches are most suitable for handling instances with this type of complexity.

In the future, the results of data mining analysis may be incorporated directly into the database web site, instead of being computed offline. This more explicit integration could allow investigators to perform computational predictions on target proteins as they are entered into the system.

Future Directions: Global Surveys

SPINE currently focuses on the front end of large-scale proteomics efforts, collecting the experimental data generated before structures have been determined. As the NESGC project matures, we anticipate that the database will incorporate more and more information about completed protein structures. The analytical theme will then shift from

optimization of high-throughput structure determination to presenting a global survey or protein folds in various genomes, similar in spirit to a number of previous studies [35,36,37,38].

Acknowledgments

The authors wish to thank Anna Khachatryan for cloning the MT genes, and Steven Beasley, Brian Le and Anthony Semesi for protein purification. The authors acknowledge support from the National Institutes of Health Protein Structure Initiative (P50 grant GM62413-01), the New Jersey Commission on Science and Technology, the Merck Genome Research Institute, and the Ontario Research and Development Challenge Fund.

References

1. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A., and Wheeler, D.L. (2000). Genbank. *Nucl Acids Res* **28**: 15-18.

2. Tateno, Y., Miyazaki, S., Ota, M., Sugawara, H., and Gojobori, T. (2000). DNA bank of Japan (DDBJ) in collaboration with mass sequencing teams. *Nucl Acids Res* **28**: 24-26.

3. Baker, W., van der Broek, A., Camon, E., Hingamp, P., Sterk, P., Stoesser, G., and Tuli, M.A. (2000). The EMBL nucleotide sequence database. *Nucl Acids Res* **28**: 19-23.

4. Barker, W.C., Garavelli, J.S., Huang, H., McGarvey, P.B., Orcutt, B., Srinivasarao, G.Y., Xiao, C., Yeh, L.S., Ledley, R.S., Janda, J.F., Pfeiffer, F., Mewes, H.W., Tsugita, A., and Wu, C. (2000). The Protein Information Resource (PIR). *Nucl Acids Res* **28**: 41-44.

5. Bairoch, A., and Apweiler, R. (2000). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucl Acids Res* **28**: 45-48.

6. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig,H., Shindyalov, I.N., and Bourne, P.E. (2000). The Protein Data Bank. *Nucl Acids Res* **28**: 235-242.

7. Cherry, J.M., Adler, C., Ball, C., Chervitz, S.A., Dwight, S.S., Hester, E.T., Jia, Y., Juvik, G., Roe, T., Schroeder, M., Weng, S., and Botstein, D. (1998). SGD: Saccharomyces Genome Database. *Nucl Acids Res* **26**:73-80.

8. Mewes, H.W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schuller, C., Stocker, S., and Weil, B. (2000). MIPS: a database for genomes and protein sequences. *Nucl Acids Res* **28**:37-40.

9. Gelbart WM, Crosby M, Matthews B, Rindone WP, Chillemi J, Russo Twombly S, Emmert D, Ashburner M, Drysdale RA, Whitfield E, Millburn GH, de Grey A, Kaufman T, Matthews K, Gilbert D, Strelets V, and Tolstoshev C (1997). FlyBase: a Drosophila database. The FlyBase consortium. *Nucl Acids Res* **25**:63-66.

10. Frishman, D., Heumann, K., Lesk, A., and Mewes, H.W. (1998). Comprehensive, comprehensible, distributed and intelligent databases: current status. *Bioinformatics* **14**: 551-61.

11. Tatusov, R.L., Koonin, E.V., and Lipman, D.J. (1997). A genomic perspective on protein families. *Science* **278**:631-637.

12. National Center for Biotechnology Information (ncbi.nlm.nih.gov/geo).

13. Aach, J., Rindone, W., and Church, G.M. (2000). Systematic management and analysis of yeast gene expression data. *Genome Res* **10**:431-45.

14. Xenarios, I., Rice, D.W., Salwinski, L., Baron, M.K., Marcotte, E.M., and Eisenberg, D. (2000). DIP: the database of interacting proteins. *Nucl Acids Res* **28**:289-91.

15. Bader, G.D. and Hogue, C.W. (2000). BIND - a data specification for storing and describing biomolecular interactions, molecular complexes and pathways. *Bioinformatics*. **16**:465-77.

16. Gerstein, M. and Krebs, W. (1998). A database of macromolecular motions. *Nucl Acids Res* **26**:4280-4290.

17. Qian, J., Stenger, B., Wilson, C.A., Lin, J., Jansen, R., Teichmann, S.A., Park, J., Krebs, W.G., Yu, H., Alexandrov, V., Echols, N., and Gerstein, M. (2001). PartsList: a web-based system for dynamically ranking protein folds based on disparate attributes,

including whole-genome expression and interaction information. *Nucl Acids Res* **29**:1750-1764.

18. Gerstein, M. (1998). Patterns of protein-fold usage in eight microbial genomes: A comprehensive structural census. *Proteins* **33**:518-534.

19. Lin, J. and Gerstein, M. (2000). Whole-genome trees based on the occurrence of folds and orthologs: Implications for comparing genomes on different levels. *Genome Res* **10**:808-818.

20. Brenner, S.E., Barken, D., and Levitt, M. (1999). The Presage database for structural genomics. *Nucl Acids Res* **27**:251-253.

21. Goodman, N., Rozen, S., and Stein, L. (1995). Labbase: A database to manage laboratory data in a large-scale genome-mapping project. *IEEE Computers in Medicine and Biology* **14**:702-709.

22. Yona, G., Linial, N., and Linial, M. (2000). ProtoMap: Automatic classification of protein sequences and hierarchy of protein families. *Nucl Acids Res* **28**:49-55.

23. Qian, J., Stenger, B., Wilson, C.A., Lin, J., Jansen, R., Krebs, W., Alexandrov, V., Echols, N., Teichmann, S., Park, J., and Gerstein, M. (2000). PartsList: a web-based system for dynamically ranking protein folds based on disparate attributes. (submitted) {{

24. Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol* **247**:536-540.

25. Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., and Thornton, J.M. (1997), CATH - A hierarchic classification of protein domain structures. *Structure* **8**:1093-1108.

26. Christendat, D., Yee, A., Dharamsi, A., Kluger, Y., Savchenko, A., Cort, J.R., Booth, V., Mackereth, C.D., Saridakis, V., Ekiel, I., Kozlov, G., Maxwell, K.L., Wu, N., McIntosh, L.P., Gehring, K., Kennedy, M.A., Davidson, A.R., Pai, E.F., Gerstein, M., Edwards, A.M., and Arrowsmith, C.H. (2000). Structural proteomics of an archeon. *Nat Struct Biol* **7**:903-909.

27. Quinlan, J.R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*. **27**: 221-234.

28. Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kauffman.

29. Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis* **1**:131-156.

30. Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA:Addison-Wesley.

31. Cortes, C. and Vapnik, V. (1995), Support vector networks. *Machine Learning* **20**:273-297.

32. Engelman, D.M., Steitz, T.A., and Goldman, A. (1986). Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Annu Rev Biophys Biophys Chem* **15**:321-353.

33. Quinlan, J.R. (1996). Bagging, boosting, and C4.5. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

34. Nadkarni, P.M., Marenco, L., Chen, R., Skoufos, E., Shepherd, G., and Miller, P. (1999). Organization of heterogeneous scientific data using the EAV/CR representation. *JAMIA* **6**:478-493.

35. Gerstein, M., Lin, J., and Hegyi, H. (2000). Protein folds in the worm genome. *Pac Symp Biocomput* 30-41.

36. Gerstein, M. and Hegyi, H. (1998). Comparing genomes in terms of protein structure: Surveys of a finite parts list. *FEMS Microbiol Rev* **22**:277-304.

37. Gerstein, M. (1998). How representative are the known structures of the proteins in a complete genome? A comprehensive structural census. *Fold Des* **3**:497-512.

38. Gerstein, M. (1997). A structural census of genomes: Comparing bacterial, eukaryotic, and archaeal genomes in terms of protein structure. *J Mol Biol* **274**:562-576.

39. Wootton, J.C. and Federhen, S. (1996). Analysis of compositionally biased regions in sequence databases. *Meth Enzymol* **266**:554-571.

40. Garnier, J., Osguthorpe, D.J., Robson, B. (1978). Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J Mol Biol* **120**:97-120.

Figure legends

Figure 1. Global project summary (A), statistics display (B), and database home page (C). The summary table can be dynamically reconfigured to present subsets of database entries, selected based on a number of simple parameters such as the target genome the protein originates from, or the institution submitting the entries. An additional parameter, labeled “Attribute”, is used to narrow the search to entries whose experimental progress corresponds to a particular chronological stage in the table. For example, entries can be selected with an attribute of “secondary structure”, which will retrieve all constructs having secondary structure data derived through various biophysical characterization methods.

Figure 2. A) Relationships between database system components; B) Software module dependencies. The system was developed using the mySQL database engine for the Linux platform, in conjunction with two programming languages to facilitate low-level database interaction and development of the user interface software: Perl 5.005 with the Perl Database Interface (DBI) module, and the PHP 3.0 hypertext preprocessor. While syntactically similar, each language features distinct capabilities. Because the PHP interpreter is integrated as an Apache web server module, execution of PHP programs is generally faster than that of Perl-based CGI programs. This makes PHP well-suited to interactive systems where timely server responses are a priority. While syntactically straightforward, the PHP language does not offer the extensive programming flexibility of Perl5. The core of the user interface system was therefore developed in PHP, while auxiliary components requiring more sophisticated functionality were implemented in Perl.

Figure 3. Core schema for the expanded database. Relational tables capture data for target proteins, their related expression constructs, and separate sets of experimental parameters for expression, purification, X-ray crystallography, NMR, and biophysical characterization. Additionally, a number of features have been developed to record lab management and transaction information (tables not shown).

Figure 4. Overwrite protection during the creation of new database records. The first step in creating a database record is assigning an identifier to the new entry. The identifier consists of three parts: a character to represent the target organism, a second character to indicate the institution from which the entry originates, and a unique alphanumerical character string. When the entry identifier is selected, the character string component may be chosen by the investigator if a proprietary nomenclature scheme is preferred; otherwise it can be automatically assigned by the system. In the latter case, the unique identifier is the next available integer following the combination of target organism and institution codes. Whether the character string component is selected by the user or generated by the system, new construct identifiers are examined by the software and guaranteed not to conflict with those of existing entries, protecting against the accidental overwriting of data. Once a valid identifier has been assigned to the new database record, the user may input relevant experimental parameter values using the construct entry form. Database records may be recalled and updated in two ways: by pressing the edit button available on its associated display page, or by entering an expression construct identifier directly into a form accessible from the main database web interface. Once a record has been selected, all of its existing field values are displayed in the construct editor, which shares a layout similar to the entry form. Users are then able to enter additional data and/or edit the current values associated with the construct, and store the updated record in the database.

Figure 5. Database searching and record retrieval. Users can construct complex Boolean searches on a number of database key fields with an intuitive form (A); the form elements are then parsed internally and an SQL query is created based on the values of the form elements and executed against the database. The search results are then summarized in a table, displaying a user-selectable number of entries per page (B). The query terms also appear above the table in a pseudo-English format, to assist in performing effective searches. Selecting an entry from the table displays the expression construct record in a separate web page (C), which contains all the database fields associated with the record, in addition to a number of links to external resources (D).

Figure 6. Conceptual structure of the decision tree model used for classification problems. Instances are sorted from root to leaf nodes, based on a number of properties defined at each node by splitting variables. Pictured is a decision tree built to predict the tendency for protein crystallization based on sequence features such as amino acid content, hydrophobicity, and homology to other sequences. The nodes of the tree are represented by ellipses; the values to the left of each node indicate the number of proteins which are unable to crystallize, while those to the right denote the crystallized examples. The splitting threshold for each node appears directly under its associated variable. The decision tree algorithm calculates all possible splitting thresholds for each variable, selecting each variable and its threshold to optimize the homogeneity of the two subsequent nodes. When a variable v is split, the right branch is assigned to $v <$ threshold, and the left branch corresponds to $v >$ threshold.

Figure 7. Decision trees built for solubility prediction. Tree pruning methods are designed to reduce the number of nodes, and arrive at the smallest tree whose error rate performance is closest to the minimal error rate of the entire tree. A, B) Uppermost levels of two decision trees, highlighting paths for classification rules. The original trees from which these subsets of nodes were derived are inset to the right. Decision tree A was built using the entire set of 562 proteins, while B was trained and tested on discrete randomized subsets of the proteomics data: 375 proteins were used for training and the remaining 187 for testing. Soluble and insoluble proteins are indicated by the numbers to the right and left of each node, respectively. In the case of decision tree B, two values are used for each class, corresponding to training (left) and testing (right) phases. Decision pathways which terminate in highly homogeneous nodes (mostly dark = soluble, mostly white = insoluble) and are not distant from the root define more robust rules which can generalize against unseen examples. Heterogeneous nodes could be further split by extending the tree downward, improving the error rate but overfitting the training set. The pathways indicated in each decision tree represent sets of rules. For instance, the right branching path of example A (indicated in green) selects mostly soluble proteins, based on the condition that the combined compositions of acidic residues (*C*(DE)) in their sequences exceed 18%. The left branching path of the same tree (in red) outlines the following set of conditions, and classifies proteins which are likely to be insoluble: *C*(DE) less than 18%; presence of a stretch of amino acids with average hydrophobicity less than -0.78 kcal/mole (labeled *Hphobe*); fewer than 16% acidic amino acids and their amides (*C*(DENQ)). C) Thresholds at which each node partitions the input vectors in the upper levels of the two decision trees. At each level, the nodes are listed sequentially from left to right (e.g., at level 2 in tree A, the leftmost node represents the splitting variable *Hphobe* having a threshold of -.78 on the GES hydrophobicity scale, followed by a node in the rightmost branch of the tree corresponding to the splitting variable *Length* with a threshold of 95 amino acids).

Table legends

Table 1. Listing of prototype database fields and their utility in data mining analysis. The level of standardization of experimental parameters is indicated by the shading of each database attribute, followed by example values and a description of each field. Darkly shaded fields were the focus of classifier training and predictions.

Table 2. Protein sequence features used for solubility prediction. Amino acid compositions and biochemical properties formed the basis of the feature set, secondary structure prediction, hydrophobicity scores on the GES scale [32], and entropic complexity measures calculated by the SEG program [39]. Long low-complexity regions were identified with SEG using the standard parameters, a trigger complexity $K(1)$ of 3.4, an extension complexity $K(2)$ of 3.75, and a sequence window of length 45. These domain sized compositionally-biased elements are often associated with non-globular parts of proteins that do not readily fold and may aggregate in solution. Short low-complexity regions were identified using a trigger complexity of $K(1) = 3.0$, an extension complexity $K(2) = 3.3$, an a window of length 25. In order to enhance predictability and model simplicity, feature selection algorithms were implemented to extract a feature subset, highlighted in braces. The experimentally determined solubility values were used in the training phase of supervised learning. Secondary structure prediction was carried out with the GOR program [40]. Signal sequences are identified via pattern matching, and contain a charged residue within first seven amino acids, followed by a stretch of 14 hydrophobic peptides (measured on the GES hydrophobicity scale).