Prediction of Subcellular Locations for Fungal Proteins

James D. Munyon[*] Xiangjia Min[†] Sepideh Khavari[‡] Guang-Hwa Chang[§]

Abstract
Proteins perform many functions within the cells of organisms, and these functions are closely related to their subcellular locations: where in a cell they reside. Protein sequences are entering databases faster than their subcellular locations can be empirically measured, so there is a need for predictors that can accurately predict protein subcellular locations. One numerical representation of the amino acid composition of proteins is called pseudo amino acid composition, turning a long string of amino acids which make up a protein into a length 20 (or greater) vector (whose first 20 values are the normalized occurrence frequencies of the 20 "standard" amino acids). Through this transformation, using a benchmark dataset of 3002 fungal proteins, initial decision tree methods of random forests, AdaBoost, SAMME, and bagging were applied to protein data to establish "baseline" predictive performance results, and then some prediction methods found in the literature, support vector machines and the covariant discriminant algorithm, were applied as well. We found that support vector machines improve over all of the decision tree methods, with the covariant discriminant algorithm giving an even further improvement, with potential room to perform better in and of itself in the near future, as long as more data can be made available for locations with small numbers of protein representatives in the benchmark dataset. Performance comparisons were also made with some computational tools which were able to accept the benchmark dataset (or subsets thereof).

Key Words: Protein subcellular location prediction, Random forests, AdaBoost, SAMME, Support vector machines, Covariant discriminant algorithm

1. Introduction

Kuo-Chen Chou and Hong-Bin Shen's 2007 review paper [1], "Recent progress in protein subcellular location prediction", provides an excellent introduction to this area of research, and both of the authors are associated with many papers in the literature. The review article provides a good basis for much of the introduction section here.

First of all, we must mention biological cells. "...the cell is deemed to be the most basic structural and functional unit of all living organisms and often is called a 'building block of life' " [1]. Many different components, probably known as organelles, perform specialized tasks inside cells. It is mostly these organelles that we will come to term as "subcellular locations". We can briefly describe the functions of some organelles, specifically those which will appear later in the paper as potential locations for proteins to be predicted to. The plasma membrane surrounds the cell as a lipid layer that controls what goes in and out of the cell. The cytoplasm is the "jelly" that holds the cell together, and other organelles are suspended in it. The cytoskeleton is a network of filaments that branches through the cytoplasm, serving many functions. The endoplasmic reticulum (ER) synthesizes proteins and lipids and transports proteins. The Golgi apparatus modifies and stores products of the endoplasmic reticulum. The mitochondria is associated with cellular respiration. The

[*]Bowling Green State University, Bowling Green, Ohio 43403-0001; jmunyon@bgsu.edu; Supplementary data, tables, etc. can be found at http://proteomics.ysu.edu/publication/data/FunPrediction_Munyon/
[†]Youngstown State University, Youngstown, Ohio 44555; xmin@ysu.edu
[‡]Youngstown State University, Youngstown, Ohio 44555; skhavari@student.ysu.edu
[§]Youngstown State University, Youngstown, Ohio 44555; gchang@ysu.edu

nucleus contains chromosomes and is basically the "powerhouse" or brain of the cell. The peroxisome turns peroxide into water. And the vacuole is a sac that holds materials such as water [2].

It is the case that many of the functions that sustain cell life are due to the work of proteins. There may be around one billion proteins in an average cell, and since these proteins are so important to cell function, it is a must to understand their functions and residencies. While physical biochemical experiments can be conducted to measure the subcellular locations of proteins, these are expensive, take time, and are impractical in the modern age of fast entry of protein data into databases. Thus, many proteins can be found in databases that either lack subcellular location annotation, or that have annotation with "uncertain labels", i.e. the annotation was not through experimental observation. If progress cannot be made, the gap between newly found proteins and their subcellular locations will grow. Quoting the paper, "To use these newly found proteins for basic research and drug discovery in a timely manner, it is highly desired to develop an effective method to bridge such a gap. During the past 15 years, a variety of predictors have been developed to deal with the challenge" [1]. The paper then goes on to discuss predictors that the authors consider distinguishable from others based on some special features.

Now that we understand the general reasons why many proteins lack subcellular location annotation and why such annotation is desired, let us shift back to understanding proteins some more. The building blocks of proteins are amino acids, and there are 20 that are found in proteins, from alanine through valine. The biological activities of a protein are mainly determined by the amino acids that constitute it, and these activities are most cell processes as well as the catalyzation of reactions in cells [3]. For the purposes of this research, the amino acid sequences of proteins will be our raw data, subject to transformation via Chou's pseudo amino acid composition (which will be described in a further section), which then will result in our cleaned-up data which will be subject to actual methods/algorithms and analysis. There also exist other ways of representing proteins as data, which can be used alone, or in conjunction with something like pseudo amino acid composition. These include full sequential representation, a protein's functional domain, GO database space [1], position-specific scoring matrices [4], and others.

In terms of our actual predictors (models), we will have models that are built with protein training data (including annotation for known subcellular locations). For then analyzing the predictive accuracy of the models, testing data (which omits known location annotation) will be fed through them, and what will result is a list of predicted locations for each protein in the testing data. These predictions are then compared with the corresponding known locations, and accuracy can be measured in a variety of ways, including by location. If a predictor has a high accuracy on the testing data, there is hope that this predictive ability will translate to other proteins out in the protein population, and that in the future, accurate predictors can actually be used by researchers to augment experimental observation of protein locations, and give reasonable predictions for proteins that truly lack subcellular location annotation.

2. Literature Review

There exist many papers in the literature on the topic of predicting protein subcellular location/localization. Since three different aspects of problem-solving exist (choosing how to represent the protein as data, choosing how to transform this raw data, and choosing prediction algorithms/methods), there exist numerous combinations of approaches for solving the problem of protein subcellular location prediction. For example, Chou and Shen's review paper focuses on the following transformation and prediction combinations: amino acid composition with the covariant discriminant algorithm (CD); pseudo amino acid composition with CD, K nearest neighbors (KNN), and optimized evidence theoretic K nearest neighbors (OET-KNN); FunD with KNN and OET-KNN; and GO with KNN and OET-KNN, where amino acid composition is a "simpler" form of pseudo amino acid composition which will be seen in a further section, FunD and GO involve proteins being coded with zeros and ones depending on "hits" against databases, the covariant discriminant algorithm is a method used in this paper, and the KNN/OET-KNN methods involve, respectively, a protein being assigned to the location that its K nearest neighbors are a part of, and a more complicated variant thereof [1]. Chou and Shen were also willing to consider the more difficult multiclass problem: a problem where proteins should be predicted to more than one location, of which many papers in the literature do not cover.

In other papers, we find many other different approaches. Some methods end up as computational tools that can be found on websites/webservers, or that can be downloaded as executables. Some of these include SignalP, WoLF PSORT, Phobius, TargetP, TMHMM, FragAnchor, and PS-Scan [5]. A large list can also be found at [43] of PSORT-family tools, as well as many other tools. In this author's opinion, however, many of these tools contain a steep learning curve inherent in their usage, as their internal methods/algorithmic steps and output can be quite cryptic and difficult for non-experts to read and interpret. Also for example, TargetP only predicts proteins to one of three possible locations, and this could be considered "not enough" or "too broad" in terms of possibilities [6]. At the end of this paper, performance results of some of these tools will be discussed briefly.

Otherwise, we can also find, as examples: the covariant discriminant algorithm predicting for apoptosis proteins [7]; pseudo amino acid composition in conjunction with the Lyapunov index, Bessel function, Chebyshev filter and complexity measure factor [8,9]; position-specific scoring matrices, principal component analysis, and support vector machines [4]; log-odds sequence logos [10] and a host of other method combinations. The coverage of just these mentioned papers shows the many different possible approaches to predicting locations of these proteins, and certainly there exist plenty more methods that still need to be attempted or created.

3. Methodology

The data for this project was obtained from the UniProt Knowledgebase (UniProtKB), in the "reviewed entries" subset, named "Swiss-Prot". Proteins from fungal organisms were the focus of this project. To define a good "benchmark" dataset, several subsetting criteria which are found in many papers in the literature were followed:

- Proteins must be reviewed and have annotation for subcellular location
- Proteins must only have annotation for one location (as the multi-class prediction problem is more difficult, and not the subject of this project)
- Protein location evidence must be experimental (not inferential)
- Proteins must not be fragments (amino acid sequences must start with methionine)
- Proteins must not have unknown amino acids anywhere in their sequences

- Proteins must have 100 or more amino acids in their sequences (this qualifies as "sufficiently long")

After subsetting by the previous rules, "50/50" BLASTClust was then applied to the proteins (if two or more proteins are at least 50% similar over at least 50% of their lengths, they are considered as part of a cluster, and only one of them is randomly chosen to be kept). After all of this subsetting, 3002 fungal proteins remained, their annotated subcellular locations being the following: cytoplasm for 770 proteins, cytoskeleton for 95 proteins, endoplasmic reticulum (ER) for 38 proteins, endoplasmic reticulum membrane (ER membrane) for 247 proteins, Golgi apparatus for 24 proteins, Golgi apparatus membrane for 75 proteins, mitochondria for 365 proteins, mitochondria membrane for 187 proteins, nucleus for 952 proteins, nuclear membrane for 24 proteins, peroxisome for 8 proteins, peroxisome membrane for 11 proteins, plasma membrane for 10 proteins, secreted for 82 proteins, vacuole for 14 proteins, and vacuole membrane for 100 proteins, for a total of 16 different subcellular locations being represented. For initial methods, the data was divided into a 70% testing set and a 30% training set; however, later methods would take advantage of cross-validation (either ten-fold or jackknife) and thus use the entire dataset for training and testing. Then the data was appropriately transformed via Chou's pseudo amino acid composition, using a correlation factor of $\lambda = 15$. This left us with a data matrix of 3002 rows and 35 columns (3002 observations of 35 variables), where 35 comes from 20 standard amino acids + 15 interaction terms.

3.1 Random Forests

From the abstract of Leo Breiman's famous 2001 paper "Random Forests" [13]: "Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest". For a slightly-modified version of Breiman's definition: "A random forest is a classifier consisting of a collection of tree-structured classifiers [using] independent identically distributed random vectors [for determining the different variables involved in the creation of each individual tree] and each tree casts a unit vote for the most popular class [for each data point]...". In simple terms, a random forest is a combination of many individual tree predictors, where each decision tree is built using a randomly-selected subset of our variables and a bootstrapped sample of our training data. Each tree then outputs a class prediction for each data point, and the mode of the predictions for a data point becomes its final prediction. Breiman's paper goes into detail and covers many topics such as theoretical justification, bounds on error rates, comparisons to adaptive boosting, implementation, etc. A quick suggestion for the success of random forests is that an upper bound for the error rate of a decision tree ensemble can be shown to depend on the correlations between the trees in the ensemble, and that a random forest can have low between-tree correlations.

3.2 Adaptive Boosting

As mentioned in [14], boosting can be a successful technique for solving a two-class classification problem (in the case of a multi-class problem, the problem is split up into several two-class problems). A well known specific algorithm is "AdaBoost" as given by Freund and Schapire in 1997. In AdaBoost, the idea is to combine many weak classifiers in an appropriate linear combination such that the resulting final classifier is very accurate. Start with all training data points equally-weighted and build a classifier (such as a decision tree).

For data points that are misclassified, increase their weights, and for points that have been correctly classified, decrease their weights. Then build a new classifier. This classifier will hopefully classify the higher-weighted data points better than the previous classifier. Repeat this process many, many times, then combine all of the classifiers in a linear combination to obtain final classifications for all training data points. Let training data be defined as $(\mathbf{x}_1, c_1), ..., (\mathbf{x}_n, c_n)$ where a vector of predictor variables $\mathbf{x}_i \in \mathbb{R}^n$ and a response variable $c_i \in 1, 2, ..., K$ (is qualitative, taking on one of a finite number of values). The AdaBoost algorithm is then as follows:

- Initialize observation weights as $w_i = 1/n, i = 1, 2, ..., n$
- For $m = 1$ to M (with M being the number of classifiers to build):
  - Fit a classifier $T^{(m)}(x)$ to the training data using weights w_i
  - Compute $err^{(m)} = \sum_{i=1}^{n} w_i \times \mathbb{I}\left(c_i \neq T^{(m)}(x_i)\right) / \sum_{i=1}^{n} w_i$
  - Compute $\alpha^{(m)} = log\left(\frac{1 - err^{(m)}}{err^{(m)}}\right)*$
  - Set $w_i \leftarrow w_i \times exp\left(\alpha^{(m)} \times \mathbb{I}\left(c_i \neq T^{(m)}(x_i)\right)\right), i = 1, 2, ..., n$
  - Re-normalize w_i
- Output $C(x) = \underset{x}{\operatorname{argmax}} \sum_{m=1}^{M} \alpha^{(m)} \times \mathbb{I}(T^{(m)}(x) = k)$

AdaBoost is considered to be very accurate for two-class problems, however, it's potential lack of applicability to multi-class problems has led to the next method to be seen.

3.3 SAMME

SAMME (Stagewise Additive Modeling using a Multi-class Exponential Loss Function) is the same as AdaBoost, with the only difference being condition * from before. In SAMME, $\alpha^{(m)} = log\left(\frac{1 - err^{(m)}}{err^{(m)}}\right) + log(K - 1)$. Note that when $K = 2$, SAMME reduces to AdaBoost. Otherwise, the one change can make SAMME a better alternative to AdaBoost in multi-class problems, with details given in [14].

3.4 Bagging

Random forests weren't Breiman's first foray into ensembles of decision trees. In 1996, his paper "Bagging Predictors" [15] introduced the idea of combining multiple predictors such that each one used a bootstrapped sample of the training data, to improve performance and the potential instability inherent in some procedures. The word "bagging" is a stand-in for "bootstrap aggregating", and the terms are used interchangeably. One of the sections in his paper deals with the application of bagging to decision trees. We can explain our implementation with simplicity. We build multiple decision trees, where each tree uses a bootstrapped sample of our training data. Thus, for an individual classifier, a data point from the training data may appear in our bootstrapped training data once, more than once, or not at all. After training our many, many trees, the final prediction for a data point is the mode of the predictions given for that data point by all of the trees.

3.5 Support Vector Machines

Some papers in the literature, including very recent papers, suggest the use of support vector machines (SVMs) as appropriate models for classification [4,16]. Note that the method only considers two-class classification problems "by default", but modifications can be made for a problem like ours that involves 16 classes (locations). By [17], the algorithm can be defined. Consider testing data points (of one of two possible classes) as labeled pairs of the form $(\mathbf{x}_i, y_i), i = 1, ..., l$ with $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$. Each $\mathbf{x}_i$ is the vector of variable values associated with a specific data point, and its associated y_i indicates which of the two classes the data point is a member of. Taken together, each $(\mathbf{x}_i, y_i)$ is one row of our testing data matrix. The following quadratic optimization problem is then solved:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ &\text{subject to } y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ &\qquad\qquad\qquad\qquad\qquad\quad \xi_i \geq 0. \end{aligned} \tag{1}$$

What we have are the training vectors $\mathbf{x}_i$ being mapped to a higher, possibly infinite dimensional space by the function ϕ. The idea is that, in a higher dimensional space, data points of different classes may be linearly separable using a hyperplane, where they weren't linearly separable originally. Or if they still aren't linearly separable, they are at least much more linearly separable than they originally were. When training points are (closely or fully) linearly separable, testing data points can then be predicted based on which side of the separating hyperplane they fall on. Back in (1), $C > 0$ and the ξ_i indicate that perfect separation may not be attained as previously mentioned. Also important is $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ (the kernel function), where we use the radial basis function (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0. \tag{2}$$

C and γ are the two parameters for the researcher to choose (or, practically, find the best pair using cross-validation). Note that our problem involves 16 possible classes, not just two, so to use the SVM method, $(16)(16-1)/2 = 120$ "one vs. one" models must be created, where each one considers just training data points of two classes and finds the best separation between just those two. A voting scheme then determines the overall final prediction for a data point.

3.6 Covariant Discriminant Algorithm

This (somewhat older) method is mentioned in [1,18], and can be described as follows by [1]. Consider $(\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N)$, a group of N proteins, from the M possible subcellular locations $(S_1, S_2, ..., S_M)$. In our problem, $N = 3002$ and $M = 16$. Now consider any location subset S_m. It's uth protein is represented as

$$P_m^u = [p_{m,1}^u p_{m,2}^u \cdots p_{m,20}^u \cdots p_{m,20+\lambda}^u]^T, \tag{3}$$

each entry being one of the $20+\lambda$ values from the protein's pseudo amino acid composition. Also define the standard vector of S_m as

$$\bar{\mathbf{P}}_m = [\bar{p}_{m,1}\bar{p}_{m,2}\cdots\bar{p}_{m,20}\cdots\bar{p}_{m,20+\lambda}]^T, \tag{4}$$

each entry being an average pseudo amino acid composition value for all of the proteins in S_m. Consider $\bar{\mathbf{P}}_m$ to be the "standard" or "average" protein in subcellular location m. Now let **P** be a query protein whose location we wish to predict. We want to consider the similarity between **P** and each $\bar{\mathbf{P}}_m$. Define our similarity measure as follows:

$$F(\mathbf{P}, \bar{\mathbf{P}}_m) = D^2_{Mah}(\mathbf{P}, \bar{\mathbf{P}}_m) + ln|\mathbf{C}_m|, \tag{5}$$

where

$$D^2_{Mah}(\mathbf{P}, \bar{\mathbf{P}}_m) = (\mathbf{P} - \bar{\mathbf{P}}_m)^T \mathbf{C}_m^{-1} (\mathbf{P} - \bar{\mathbf{P}}_m) \tag{6}$$

is the squared Mahalanobis distance between **P** and $\bar{\mathbf{P}}_m$. $\mathbf{C}_m$ is the $(20 + \lambda) \times (20 + \lambda)$ covariance matrix for S_m with entry

$$c^m_{i,j} = \frac{1}{N_m - 1} \sum_{u=1}^{N_m} \left(p^u_{m,i} - \bar{p}_{m,i}\right) \left(p^u_{m,j} - \bar{p}_{m,j}\right), \tag{7}$$

$\mathbf{C}_m^{-1}$ is its inverse, $|\mathbf{C}_m|$ is its determinant, and N_m is just the number of proteins in S_m. It should be noted that due to properties of the Pseudo Amino Acid Composition, any covariance matrix $\mathbf{C}_m$ will be singular and hence not have an inverse. The work-around will be the following "dimension-reducing" procedure: drop one of the $20 + \lambda$ variables (normalized amino acid frequencies or interaction terms) so that only $20 + \lambda - 1$ are considered. Then the covariance matrices become theoretically invertible (although it may still be difficult in practice), and progress can continue. It can also be shown that it doesn't matter which variable is dropped: $F(\mathbf{P}, \bar{\mathbf{P}}_m)$ values will end up the same no matter which variable is dropped [1]. In our case, $M = 16$ $F(\mathbf{P}, \bar{\mathbf{P}}_m)$ values will be computed for query protein **P**, and the location associated with the smallest one will become the location prediction for **P**.

4. Analysis

In the spirit of [19], all of our decision tree methods were run using either the data split into a 70% training set and a 30% testing set, or using ten-fold cross-validation on the entire dataset. In the training set of 2101 proteins, locations were as follows: cytoplasm for 526 proteins, cytoskeleton for 66 proteins, endoplasmic reticulum (ER) for 29 proteins, endoplasmic reticulum membrane (ER membrane) for 174 proteins, Golgi apparatus for 18 proteins, Golgi apparatus membrane for 50 proteins, mitochondria for 268 proteins, mitochondria membrane for 132 proteins, nucleus for 663 proteins, nuclear membrane for 17 proteins, peroxisome for 6 proteins, peroxisome membrane for 7 proteins, plasma membrane for 5 proteins, secreted for 58 proteins, vacuole for 9 proteins, and vacuole membrane for 73 proteins, and in the testing set of 901 proteins, locations were as follows: cytoplasm for 244 proteins, cytoskeleton for 29 proteins, endoplasmic reticulum (ER) for 9 proteins, endoplasmic reticulum membrane (ER membrane) for 73 proteins, Golgi apparatus for 6 proteins, Golgi apparatus membrane for 25 proteins, mitochondria for 97 proteins, mitochondria membrane for 55 proteins, nucleus for 289 proteins, nuclear membrane for 7 proteins, peroxisome for 2 proteins, peroxisome membrane for 4 proteins, plasma membrane for 5 proteins, secreted for 24 proteins, vacuole for 5 proteins, and vacuole membrane for 27 proteins. All analysis was done using R statistical software, except for the attempted calculation of some matrix inverses, which was done in MATLAB (to no avail).

4.1 Random Forests

R's "randomForest" package contains implementations of Leo Breiman's original random forest algorithm. We built a forest of size 500 trees on our 70% training data and then used the model to predict subcellular locations for the 30% testing data. Recall that a random selection of variables is used to build each tree.

Table 1: Statistics by Class

	Sn	Sp	PPV	NPV	Pr	DR	DP	BA	MCC	NObs
Cytoplasm	0.45	0.76	0.41	0.79	0.27	0.12	0.3	0.6	0.2	244
Cytoskeleton	0	1	NA	0.97	0.03	0	0	0.5	0	29
ER	0	1	NA	0.99	0.01	0	0	0.5	0	9
ER (membrane)	0.52	0.95	0.47	0.96	0.08	0.04	0.09	0.73	0.45	73
Golgi apparatus	0	1	NA	0.99	0.01	0	0	0.5	0	6
Golgi apparatus (membrane)	0.04	1	1	0.97	0.03	0	0	0.52	0.2	25
Mitochondria	0.32	0.97	0.53	0.92	0.11	0.03	0.07	0.64	0.36	97
Mitochondria (membrane)	0.04	1	1	0.94	0.06	0	0	0.52	0.18	55
Nucleus	0.78	0.61	0.48	0.86	0.32	0.25	0.52	0.69	0.36	289
Nucleus (membrane)	0	1	NA	0.99	0.01	0	0	0.5	0	7
Peroxisome	0	1	NA	1	0	0	0	0.5	0	2
Peroxisome (membrane)	0	1	NA	1	0	0	0	0.5	0	4
Plasma membrane	0	1	NA	0.99	0.01	0	0	0.5	0	5
Secreted	0.62	0.99	0.65	0.99	0.03	0.02	0.03	0.81	0.63	24
Vacuole	0	1	NA	0.99	0.01	0	0	0.5	0	5
Vacuole (membrane)	0	1	0	0.97	0.03	0	0	0.5	-0.01	27

Key:
Sn: Sensitivity, Sp: Specificity, PPV: Positive Predicted Value, NPV: Negative Predicted Value, Pr: Prevalence, DR: Detection Rate, DP: Detection Prevalence, BA: Balanced Accuracy, MCC: Matthews Correlation Coefficient, NObs: Number of Proteins

Table 1 shows statistics by class for the predictive ability of the random forest model on the training data. Here and elsewhere, covariance matrices will be omitted in the interest of avoiding difficult-to-read/-understand tables. It was the case that 47% of proteins in the training data were correctly predicted to their subcellular locations. Unfortunately for many locations, by sensitivities of 0 and specificities of 1, it is the case that no proteins are predicted to those locations, leaving less than half of the locations to absorb all of the predictions. By this, MCC values are 0 for those locations, which indicates a performance on par with that of random guessing. Stand-out locations for decent performance are cytoplasm, ER (membrane), mitochondria, nucleus, and secreted, in terms of balanced accuracy and MCC. Our best location is secreted: 66% sensitivity (about two-thirds correctly predicted), 99% specificity (virtually no incorrect predictions of secreted), balanced accuracy of 81%, and MCC of .63. The model performs alright but is too general to predict for locations that (for the most part) have few representatives in the testing data and overall. However, this was our best performance of all of the decision tree ensemble methods, and table 1 is a good representative for the trends that were observed in "statistics by class" tables for all of the other decision tree ensemble methods, thus those tables can be omitted to save space.

4.2 Adaptive Boosting

R's "adabag" package contains implementations of Freund and Schapire's AdaBoost, Zhu's SAMME, and Breiman's bagging. For AdaBoost to stay consistent with random forest, we build an ensemble of 500 trees using a bootstrapped sample of our training data for each tree. Recall that data point weights will not be uniform after the first tree is created. Here, 43% of proteins were correctly predicted.

4.3 Ten-fold cross-validation

For an alternative validation method, we use ten-fold cross-validation on the whole dataset instead of using the training and testing set. This allows each protein to be training data for 90% of the time, while still getting a prediction. Base, simple intuition might suggest that we should expect better results with the cross-validation. Here, 42% of proteins were correctly predicted, a surprise since cross-validation is often employed to increase predictive accuracy.

4.4 SAMME

Recall that SAMME is supposed to be a better alternative to (modified to handle more than two-class cases) AdaBoost. Implementation is the same: 70/30 training/testing split, 500 trees built on bootstrapped samples of the training data. Here, 42% of proteins were correctly predicted.

4.5 Support Vector Machines

For this method, jackknife validation is used to assess model performance. In jackknife validation, build a predictive model using all but one of your data points, and then use this model to predict the class for the one left-out data point. Repeat this process for each data point being used as the left-out data point. In our case, 3002 models are created as our benchmark dataset has 3002 proteins. Jackknife validation allows every protein to have its location predicted where the greatest possible amount of information builds the model that is doing the predicting.

Table 2: Statistics by Class

	Sn	Sp	PPV	NPV	Pr	DR	DP	BA	MCC	NObs
Cytoplasm	0.53	0.74	0.41	0.82	0.26	0.14	0.33	0.64	0.25	770
Cytoskeleton	0.01	1	0.33	0.97	0.03	0	0	0.5	0.05	95
ER	0	1	NA	0.99	0.01	0	0	0.5	0	38
ER (membrane)	0.52	0.96	0.56	0.96	0.08	0.04	0.08	0.74	0.5	247
Golgi apparatus	0	1	NA	0.99	0.01	0	0	0.5	0	24
Golgi apparatus (membrane)	0.05	1	0.44	0.98	0.02	0	0	0.53	0.15	75
Mitochondria	0.45	0.95	0.56	0.93	0.12	0.05	0.1	0.7	0.44	365
Mitochondria (membrane)	0.24	0.98	0.48	0.95	0.06	0.01	0.03	0.61	0.31	187
Nucleus	0.72	0.72	0.54	0.85	0.32	0.23	0.42	0.72	0.41	952
Nucleus (membrane)	0	1	NA	0.99	0.01	0	0	0.5	0	24
Peroxisome	0	1	NA	1	0	0	0	0.5	0	8
Peroxisome (membrane)	0.36	1	1	1	0	0	0	0.68	0.6	11
Plasma membrane	0	1	NA	1	0	0	0	0.5	0	10
Secreted	0.8	0.99	0.75	0.99	0.03	0.02	0.03	0.9	0.77	82
Vacuole	0	1	NA	1	0	0	0	0.5	0	14
Vacuole (membrane)	0.17	0.99	0.47	0.97	0.03	0.01	0.01	0.58	0.27	100

We can already see a noticeable increase in improvement over the decision tree methods, with the majority of locations seeing some kind of representation, and at least nucleus and secreted proteins performing "well".

4.6 Covariant Discriminant Algorithm

For this method we also use jackknife validation, as it lends itself to the method very nicely. For each protein, "simply" compare it to the average protein of each location using the previously-mentioned similarity measure (5), which involves a non-Euclidean measure of distance. The location associated with the smallest similarity measure value for a protein becomes that protein's location prediction.

Table 3: Statistics by Class

	Sn	Sp	PPV	NPV	Pr	DR	DP	BA	MCC	NObs
Cytoplasm	0.47	0.9	0.61	0.83	0.26	0.12	0.2	0.68	0.4	770
Cytoskeleton	0.92	0.94	0.34	1	0.03	0.03	0.09	0.93	0.54	95
ER	1	1	1	1	0.01	0.01	0.01	1	1	38
ER (membrane)	0.7	0.96	0.62	0.97	0.08	0.06	0.09	0.83	0.63	247
Golgi apparatus	0	1	NA	0.99	0.01	0	0	0.5	0	24
Golgi apparatus (membrane)	0.92	0.98	0.51	1	0.02	0.02	0.04	0.95	0.68	75
Mitochondria	0.65	0.94	0.61	0.95	0.12	0.08	0.13	0.8	0.58	365
Mitochondria (membrane)	0.78	0.94	0.47	0.98	0.06	0.05	0.1	0.86	0.57	187
Nucleus	0.57	0.91	0.74	0.82	0.32	0.18	0.24	0.74	0.52	952
Nucleus (membrane)	0	1	NA	0.99	0.01	0	0	0.5	0	24
Peroxisome	0	1	NA	1	0	0	0	0.5	0	8
Peroxisome (membrane)	0	1	NA	1	0	0	0	0.5	0	11
Plasma membrane	0	1	NA	1	0	0	0	0.5	0	10
Secreted	0.99	1	0.93	1	0.03	0.03	0.03	0.99	0.96	82
Vacuole	0	1	NA	1	0	0	0	0.5	0	14
Vacuole (membrane)	0.93	0.97	0.49	1	0.03	0.03	0.06	0.95	0.66	100

Notice our new best classification accuracy of 61%. Also of note are ER proteins being perfectly predicted, five locations having balanced accuracies above 90%, and all locations being more than trivially represented, with the exception of six locations. These locations have fewer observations than the number of variables being considered ($35 - 1 = 34$), thus their respective $\mathbf{C}_m$ matrices aren't invertible. The simple addition of new observations to give those six locations more than 35 observations would immediately remove this issue, and based on the fact that locations here with smaller numbers of locations (as long as they are greater than 35) perform well, we would expect great performance on these six locations as well, leading to an even larger overall classification accuracy, certainly greater than 61%.

5. Comparisons with Computational Tools and Conclusions

Sufficiently many methods have been attempted to facilitate discussion relating to the predictive strength of methods on proteins of different locations. For our decision tree methods, predictive performance is simply not competitive with the later methods, in terms of overall classification accuracy, and ability on a wide variety of locations. The methods only even "guaranteed" predictions for five locations (cytoplasm, ER membrane, mitochondria, nucleus, and secreted) at all, mainly in the top locations in terms of number of observations found in either the testing data, or entire dataset in the cross-validation cases. In any particular case, the method was really only notably accurate on nucleus proteins as well, which helped to raise the classification accuracy, but didn't allow the model to generalize to other locations.

Our later methods (support vector machines and the covariant discriminant algorithm) fared better, particularly the latter, which itself would certainly perform better with just the addition of some more data points in the appropriate locations. Notice how support vector machines performs particularly well on locations with many observations (nucleus, cytoplasm, etc.), where the covariant discriminant algorithm performs particularly well on locations with fewer (but greater than 35) observations (ER, secreted, etc.).

Some comparisons of these results to those given by some computational tools were performed to the extent that comparisons can reasonably be made. Tools mainly from [43] were considered, and were as follows: SubLoc, ESLPred, HSLPred, pSLIP, Protein Prowler, TargetP, WoLF PSORT, Proteome Analyst, SLP-Local, and PredSL. The following points summarize these findings:

- Many tools only predict to three to five common locations, where these methods can predict to any locations that are found in the benchmark dataset.
- Some tools' websites were "down".
- WoLF PSORT could only handle about 10% of the benchmark dataset.
- SLP-Local gave "quasi-accurate" predictions, but one location was listed as "Cytoplasm or Nucleus", which is unspecific as a prediction and superficially ensures many "technically correct" predictions.
- PredSL predicted most proteins as "Other", a very vague prediction.

The main point is that apples-to-apples comparisons were very hard to make, as all of the tested tools clearly had many uses, but not necessarily in simple, accurate location predictions of many different proteins. It seems reasonable to suggest that methods like these here in this paper should at least be used in tandem with the other important functions that the computational tools provide, as the ability to predict to more than just a preset number of locations is very important: let the data provide as much flexibility in predictions as possible.

REFERENCES

1 Chou, K.C. & Shen, H.B. (2007). Recent progress in protein subcellular location prediction. Analytical Biochemistry, 370, 1-16.

2 List of Organelles. https://bioh.wikispaces.com/List+of+Organelles.

3 (2003). The Chemistry of Amino Acids. http://www.biology.arizona.edu/biochemistry/problem_sets/aa/aa.html.

4 Yao, Y.H., Shi, Z.X. & Dai, Q. (2014). Apoptosis Protein Subcellular Location Prediction Based on Position-Specific Scoring Matrix. Journal of Computational and Theoretical Nanoscience, 11, 2073-2078.

5 Meinken, J., Asch, D.K., Neizer-Ashun, K.A., Chang, G.H., Cooper Jr, C.R. & Min, X.J. (2014). FunSecKB2: a fungal protein subcellular location knowledgebase. Computational Molecular Biology, 4(7), 1-17.

6 (2005). Output format. http://www.cbs.dtu.dk/services/TargetP-1.1/output.php.

7 Zhou, G.P. & Doctor, K. (2003). Subcellular Location Prediction of Apoptosis Proteins. Proteins: Structure, Function, and Genetics, 50, 44-48.

8 Gao, Y., Shao, S., Xiao, X., Ding, Y., Huang, Y., Huang, Z. & Chou, K.C. (2005). Using pseudo amino acid composition to predict protein subcellular location: Approached with Lyapunov index, Bessel function, and Chebyshev filter. Amino Acids, 28, 373-376.

9 Xiao, X., Shao, S., Ding, Y., Haung, Z., Haung, Y. & Chou, K.C. (2005). Using complexity measure factor to predict protein subcellular location. Amino Acids, 28, 57-61.

10 Yu, Y.K., Capra, J.A., Stojmirovic, A., Landaman, D. & Altschul, S.F. (2015). Log-odds sequence logos. Bioinformatics, 31(3), 324-331.

11 Quinlan, J.R. (1986). Induction of Decision Trees. Machine Learning, 1, 81-106.

12 CART Algorithm, retreived from ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/14.0/TREE-CART.pdf, 9:54 PM, 3/24/2015.

13 Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

14 Zhu, J., Rosset, S., Zou, H. & Hastie. T. (2006). Multi-class AdaBoost. Unpublished manuscript.

15 Breiman, L. (1996). Bagging Predictors. Machine Learning, 24, 123-140.

16 Hua, S. & Sun, Z. (2001). Support vector machine approach for protein subcellular localization prediction. Bioinformatics, 17(8), 721-728.

17 Hsu, C.W., Chang, C.C. & Lin, C.J. (2010). A Practical Guide to Support Vector Classification. Unpublished manuscript.

18 Chou, K.C. & Elrod, D.W. (1999). Protein subcellular location prediction. Protein Engineering, 12, 107-108.

19 Neizer-Ashun, K.A. (2013). Prediction of Plant Protein Subcellular Locations. Unpublished manuscript.

20 R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

21 Charif, D. & Lobry, J.R. (2007). seqinr: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis.

22 Xiao, N., Xu, Q.S. & Cao, D.S. (2014). protr: Generating Various Numerical Representation Schemes of Protein Sequence. R package version 0.5-1.

23 Hong, L. BioSeqClass: Classification for Biological Sequences. R package version 1.24.0.

24 Liaw, A & Wiener, M. (2002). Classification and Regression by randomForest. R News, 2(3), 18-22.

25 Bates, D., Maechler, M., Bolker, B. & Walker, S. (2014). lme4: Linear mixed-effects models using Eigen and S4. R package version 1.1-7, ¡URL: http://CRAN.R-project.org/package=lme4¿.

26 Kuhn, M. (2015). caret: Classification and Regression Training. R package version 6.0-41. http://CRAN.R-project.org/package=caret.

27 Alfaro, E., Gamez, M. & Garcia, N. (2013). adabag: An R Package for Classification with Boosting and Bagging. Journal of Statistical Software, 54(2), 1-35. URL http://www.jstatsoft.org/v54/i02/.

28 Ripley, B. (2014). tree: Classification and regression trees. R package version 1.0-35. http://CRAN.R-project.org/package=tree.

29 Therneau, T., Atkinson, B. & Ripley, B. (2015). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-9. http://CRAN.R-project.org/package=rpart.

30 Torgo, L. (2010). Data Mining with R, learning with case studies Chapman and Hall/CRC. URL: http://www.dcc .fc.up.pt/ ltorgo/DataMiningWithR.

31 Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. & Leisch, F. (2014). e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.6-4. http://CRAN.R-project.org/package=e1071.

32 Tang, S., Li, T., Cong, P., Xiong, W., Wang, Z. & Sun, J. (2013). PlantLoc: an accurate web server for predicting plant protein subcellular localization by substantiality motif. Nucleic Acids Research, 41, 441-447.

33 Min, X.J. (2010). Evalutation of Computational Methods for Secreted Protein Prediction in Different Eukaryotes. Journal of Proteomics & Bioinformatics, 3(4), 143-147.

34 Niu, B., Jin, Y.H., Feng, K.Y., Lu, W.C., Cai, Y.D. & Li, G.Z. (2008). Using AdaBoost for the prediction of subcellular location of prokaryotic and eukaryotic proteins. Molecular Diversity, 12, 41-45.

35 Chou, K.C. & Shen, H.B. (2010). A New Method for Predicting the Subcellular Localization of Eukaryotic Proteins with Both Single and Miltiple Sites: Euk-mPLoc 2.0. PLos ONE 5(4): e9931. doi:10.1371/journal.pone.0009931.

36 Meinken, J. & Min, X.J. (2012). Computational Prediction of Protein Subcellular Locations in Eukaryotes: an Experience Report. Computational Molecular Biology, 2(1), 1-7.

37 Neizer-Ashun, K.A., Yu, F., Meinken, J., Min, X. & Chang, G.H. Prediction of Plant Protein Subcellular Locations. Unpublished manuscript.

38 Li, H. (2013). Using the BioSeqClass Package. Unpublished manuscript.

39 Nathan, M. A Multi-site Subcellular Localizer for Fungal Proteins. Unpublished manuscript.

40 Meyer, D. (2014). Support Vector Machines - The Interface to libsvm in package e1071. Unpublished manuscript.

41 Min, X.J. (2010). Evalutation of Computational Methods for Secreted Protein Prediction in Different Eukaryotes. Journal of Proteomics & Bioinformatics, 3(4), 143-147.

42 (2003). Dr. Margaret Oakley Dayhoff. http://www.biology.arizona.edu/biochemistry/problem_sets/aa/Dayhoff .html.

43 PSORT. http://www.psort.org/.