Protein (Multi-)Location Prediction: Using Location Inter-Dependencies in a Probabilistic Framework

Ramanuja Simha and Hagit Shatkay

1 Department of Computer and Information Sciences, University of Delaware, Newark, DE, USA
2 Center for Bioinformatics and Computational Biology, DBI, University of Delaware, Newark, DE, USA
3 School of Computing, Queen’s University, Kingston, ON, Canada

Abstract. Knowing the location of a protein within the cell is important for understanding its function, role in biological processes, and potential use as a drug target. Much progress has been made in developing computational methods that predict single locations for proteins, assuming that proteins localize to a single location. However, it has been shown that proteins localize to multiple locations. While a few recent systems have attempted to predict multiple locations of proteins, they typically treat locations as independent or capture inter-dependencies by treating each locations-combination present in the training set as an individual location-class. We present a new method and a preliminary system we have developed that directly incorporates inter-dependencies among locations into the multiple-location-prediction process, using a collection of Bayesian network classifiers. We evaluate our system on a dataset of single- and multi-localized proteins. Our results, obtained by incorporating inter-dependencies are significantly higher than those obtained by classifiers that do not use inter-dependencies. The performance of our system on multi-localized proteins is comparable to a top performing system (YLoc+), without restricting predictions to be based only on location-combinations present in the training set.

1 Introduction

Knowing the location of a protein within the cell is essential for understanding its function, its role in biological processes, as well as its potential role as a drug target [1]. Experimental methods for protein localization such as those based on mass spectrometry [4] or green fluorescence detection [5], although often used in practice, are time consuming and typically not cost-effective for high-throughput localization. Hence, an ongoing effort is put into developing high-throughput computational methods [7-11] to obtain proteome-wide location predictions.

Over the last decade, there has been significant progress in the development of computational methods that predict a single location per protein. The focus on single-location prediction is driven both by the data available in public...
databases such as UniProt, where proteins are typically assigned a single location, as well as by an (over-)simplifying assumption that proteins indeed localize to a single location. However, proteins do localize to multiple compartments in the cell [12][13][14][15], and translocate from one location to another [16]. Identifying the multiple locations of a protein is important because translocation can serve some unique functions. For instance, GLUT4, an insulin-regulated glucose transporter, which is stored in the intracellular vesicles of adipocytes, translocates to the plasma membrane in response to insulin [17][18]. As proteins do not localize at random and translocations happen between designated interdependent locations, we hypothesize that modeling such inter-dependencies can help in predicting protein locations. Thus, we aim to identify associations or inter-dependencies among locations and leverage them in the process of predicting locations for proteins.

Several methods have been recently suggested for predicting multiple locations for proteins. ngLOC [19] uses a Naïve Bayes classifier to obtain independent predictions for each single location and combines these individual predictions to obtain a multi-location prediction. Li et al. [20] construct multiple binary classifiers, each using an ensemble of $k$-nearest neighbor and SVM, where each binary classifier distinguishes between a pair of locations. The predictions from all the classifiers are combined to obtain a multi-location prediction. iLoc-Euk [21] uses a multi-label $k$-nearest neighbor classifier to predict multiple locations for proteins. Similar methods were used for localizing subsets of eukaryotic proteins [22][23], virus proteins [24], and bacterial proteins [25][26]. In contrast to the machine learning-based approaches listed above, KnowPred [27] uses sequence similarity to associate proteins with multiple locations.

Notably, none of the above methods for predicting multiple locations utilizes inter-dependencies among locations in the prediction process. All the above models independently predict each single location and thus do not take into account predictions for other locations. IMMML [28] attempts to make use of correlation among pairs of locations, a simple type of dependency, when predicting multiple locations for proteins. This system does not account for more complex inter-dependencies and was not tested on any extensive protein multi-localization dataset. YLoc+ [29], a comprehensive system for protein location prediction, uses a naive Bayes classifier (see e.g. [30]) and captures protein localization to multiple locations by explicitly introducing a new class for each combination of locations supported by the training set (i.e. having proteins localized to the combination). Thus, each prediction performed by the naïve Bayes classifier can assign a protein to only those combinations of locations included in the training data. To produce its output, YLoc+ transforms the prediction into a multinomial distribution over the individual locations. We also note that as the number of possible location-combinations is exponential in the number of locations, training the naïve Bayes classifier in this manner does not provide a practical model in the general case of multi-localized proteins, beyond the training set. The performance of YLoc+ was evaluated using an extensive dataset [29] and is the highest among current multi-location predictors.
In this paper, we present a new method that directly models inter-dependencies among locations and incorporates them into the process of predicting locations for proteins. Our system is based on a collection of Bayesian network classifiers [31]. Each Bayesian Network (BN) related to each classifier corresponds to a single location $L$. Each such network is used to assign a probability for a protein to be found at location $L$, given both the protein’s features and information regarding the protein’s other possible locations. Learning each BN involves learning the dependencies among the other locations that are primarily related to proteins localizing to location $L$. For each Bayesian network classifier, its corresponding BN is learnt with the goal to improve the classifier’s prediction quality. The formulation of multi-location prediction as classification via Bayesian networks, as well as the network model are presented in Section 2. Notably, our system does not assume that all proteins it classifies are multi-localized, but rather more realistically, that proteins may be assigned to one or more locations.

We train and test our preliminary system on a dataset containing single- and multi-localized proteins previously used in the development and testing of the YLoc$^+$ system [29], which includes the most comprehensive collection of multi-localized proteins currently available, derived from the DBMLoc dataset [13]. As done in other studies [10,11,29,32], we use multiple runs of 5-fold cross-validation. The results clearly demonstrate the advantage of using location inter-dependencies. The $F_1$ score of 81% and overall accuracy of 76% obtained by incorporating inter-dependencies are significantly higher than the corresponding values obtained by classifiers that do not use inter-dependencies. Also, while our system retains a level of performance comparable to that of YLoc$^+$ on the same dataset, we note that unlike YLoc$^+$, by training the individual classifiers to predict individual — although inter-dependent — locations, the training of our system is not restricted to only those combinations of locations present in the dataset, thus our system is generalizable to multi-locations beyond those included in the training set.

The rest of the paper proceeds as follows: Section 2 formulates the problem of protein subcellular multi-location prediction and briefly provides background on Bayesian networks and relevant notations. Section 3 discusses the structure, parameters, and inter-dependencies comprising our Bayesian network collection, and introduces the learning procedure used for finding them. Section 4 presents details of the dataset, the performance evaluation measures, and experimental results. Section 5 summarizes our findings and outlines future directions.

2 Problem Formulation

As is commonly done in the context of classification, and protein-location classification in particular [8,11,12,29,33], we represent each protein, $P$, as a weighted feature vector, $f^P=(f_1^P, \ldots, f_d^P)$, where $d$ is the number of features. We view each feature as a random variable $F_i$ representing a characteristic of a protein, such as the presence or absence of a short amino acid motif [8,12], the relative abundance of a certain amino acid as part of amino-acid composition [19],.
or the annotation by a Gene Ontology (GO) term \[34\]. Each vector-entry, \( f_i^P \), corresponds to the value taken by feature \( F_i \) with respect to protein \( P \). In the experiments described here, we use the exact same representation used by Briese-meister et al. \[29\] as explained in Section 4.1.

We next introduce notations relevant to the representation of a protein’s localization. Let \( S = \{ s_1, \ldots, s_q \} \) be the set of \( q \) possible subcellular components in the cell. For each protein \( P \), we represent its location(s) as a vector of 0/1 values indicating the protein’s absence/presence, respectively, in each subcellular component. The location-indicator vector for protein \( P \) is thus a vector of the form: \( l^P = (l_1^P, \ldots, l_q^P) \) where \( l_i^P = 1 \) if \( P \) localizes to \( s_i \) and \( l_i^P = 0 \) otherwise. As with the feature values, each location value, \( l_i^P \) is viewed as the value taken by a random variable, where for each location, \( s_i \), the corresponding random variable is denoted by \( L_i \). Given a dataset consisting of \( m \) proteins along with their location vectors, we denote the dataset as: \( D = \{ (P_j, l^P_j) \mid 1 \leq j \leq m \} \).

We thus view the task of protein subcellular multi-location prediction as that of developing a classifier (typically learned from a dataset \( D \) of proteins whose locations are known) that given a protein \( P \) outputs a \( q \)-dimensional location-indicator vector that represents \( P \)'s localization.

As described in Section 1, most recent approaches that extend location-prediction beyond a single location (e.g. KnowPred \[27\], and Euk-mPLoC 2.0 \[35\]), do not consider inter-dependencies among locations. YLoc\[29\] indirectly considers these inter-dependencies by creating a class for each location-combination. Our underlying hypothesis, which is supported by the experiments and the results presented here, is that capturing location inter-dependencies directly can form the basis for a generalizable approach for location-prediction. The training of a classifier for protein multi-location prediction involves learning these inter-dependencies so that the classifier can leverage them in the prediction process. We use Bayesian networks to model such inter-dependencies.

In order to develop a protein subcellular multi-location predictor, we propose to develop a collection of classifiers, \( C_1, \ldots, C_q \), where the classifier \( C_i \) is viewed as an “expert” responsible for predicting the 0/1 value, \( l_i^P \), indicating \( P \)'s non-localization or localization to \( s_i \). In order to make use of location inter-dependencies, each \( C_i \) uses estimates of location indicators of \( P, \hat{l}_1^P, \ldots, \hat{l}_i^P, \), along with the feature-values of \( P \), in order to calculate a prediction. We use support vector machines (SVMs) (see e.g. \[30\]) to compute these estimates. The output of \( C_i \) for a protein \( P \) is given by

\[
C_i(P) = \begin{cases} 
1 & \text{If } \Pr(l_i^P = 1 \mid P, \hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P) > 0.5; \\
0 & \text{Otherwise.}
\end{cases}
\]  

Further details about the estimation procedure itself are provided in Section 3.2.

Bayesian networks have been used before in many biological applications (e.g. \[36\]\[37\]\[38\]). In this paper, we use them to model inter-dependencies among subcellular locations, as well as among protein-features and locations. We briefly introduce Bayesian networks here, along with the relevant notations (see \[39\] for more details). A Bayesian network consists of a directed acyclic graph \( G \),
Fig. 1. An example of a collection of Bayesian network classifiers we learn. The collection consists of several classifiers $C_1, \ldots, C_q$, one for each of the $q$ subcellular locations. Directed edges represent dependencies between the connected nodes. We note that there are edges among location variables ($L_1, \ldots, L_q$), as well as between feature variables ($F_1, \ldots, F_d$) and location variables ($L_1, \ldots, L_q$), but not among the feature variables. The latter indicates independencies among features, as well as conditional independencies among features given the locations.

whose nodes are random variables, which in our case represent features, denoted $F_1, \ldots, F_d$, and location indicators, denoted $L_1, \ldots, L_q$. We assume here that all the feature values are discrete. To ensure that, we use the recursive minimal entropy partitioning technique [40] to discretize the features; this technique was also used in the development of YLoc+ [29].

Directed edges in the graph indicate inter-dependencies among the random variables. Thus, as demonstrated in Figure 1, edges are allowed to appear between feature- and location-nodes, as well as between pairs of location-nodes in the graph. Edges between location-nodes directly capture the inter-dependencies among locations. We note that there are no edges between feature-nodes in our model, which reflects an assumption that features are either independent of each other or conditionally independent given the locations. This simplifying assumption helps speed up the process of learning the network structure from the data, while the other allowed inter-dependencies still enable much of the structure of the problem to be captured (as demonstrated in the results). Further details about the learning procedure itself are provided in Section 3.1.

To complete the Bayesian network framework, each node $v \in \{F_1, \ldots, F_d, L_1, \ldots, L_q\}$ in the graph is associated with a conditional probability table, $\theta_v$, containing the conditional probabilities of the values the node takes given its parents’ values, $Pr(v \mid Pa(v))$. We denote by $\Theta$ the set of all conditional probability tables, and the Bayesian network is the pair $(G, \Theta)$. A consequence of using the Bayesian network structure, is that it represents certain conditional independencies among non-neighboring nodes [39], such that the joint distribution of the set of network variables can be simply calculated as:

$$Pr(F_1, \ldots, F_d, L_1, \ldots, L_q) = \prod_{i=1}^d Pr(F_i \mid Pa(F_i)) \prod_{j=1}^q Pr(L_j \mid Pa(L_j)). \quad (2)$$

Figure 1 shows an example of a collection of Bayesian network classifiers. The collection consists of Bayesian network classifiers $C_1, \ldots, C_q$, one for each of the $q$ subcellular locations $s_1, \ldots, s_q$, where each classifier $C_i$ consists of the graph $G_i$ and its set of parameters $\Theta_i$. In each classifier $C_i$, the location indicator variable
$L_i$ is the variable we need to predict and is therefore viewed as \textit{unobserved}, and is shown as an unshaded node in the figure. The feature variables $F_1, \ldots, F_d$ are given for each protein and as such are viewed as known or \textit{observed}, shown as shaded nodes in the figure. Finally, the values for the location indicator variables for all locations except for $L_i$, $\{L_1, \ldots, L_q\} - \{L_i\}$, are needed for calculating the predicted value for $L_i$ in the classifier $C_i$. As such, they are viewed by the classifier as though they are \textit{observed}. Notably, the values of these variables are not known and still need to be estimated.

Thus, the structure and parameters of the network for each classifier $C_i$ (learnt as described in Section 3.1), are used to predict the value of each unobserved variable, $L_i$. The task of each classifier $C_i$, is to predict the value of the variable $L_i$ given the values of all other variables $F_1, \ldots, F_d$, and $\{L_1, \ldots, L_q\} - \{L_i\}$. Since, as noted above, the values of the location indicator variables $L_j$ ($j \neq i$) are unknown at the point when $L_i$ needs to be calculated, we estimate their values, using simple SVM classifiers as described in Section 3.1. We note that other methods, such as expectation maximization, can be used to estimate all the hidden parameters, which we shall do in the future.

3 Methods

As our goal is to assign locations (possibly multiple) to proteins, we use a collection of Bayesian network classifiers, where each classifier $C_i$, predicts the value (0 or 1) of a single location variable $L_i$ – while using estimates of all the other location variables $L_j$ ($j \neq i$), which are assumed to be known, as far as the classifier $C_i$ is concerned. The estimates of the location values $L_j$ are calculated using SVM classifiers as described in Section 3.1. The individual predictions from all the classifiers are then combined to produce a multi-location prediction. For each location $s_i$, a Bayesian network classifier $C_i$ must be learned from training data before it can be used. As described in Section 2, each classifier $C_i$ consists of a graph structure $G_i$ and a set of conditional probability parameters, $\Theta_i$, that is: $C_i = (G_i, \Theta_i)$. Thus, our first task is to learn the individual classifiers, i.e. their respective Bayesian network structures and parameters. The individual networks can then be used to predict a protein’s localization to each location.

Given a protein $P$, each classifier $C_i$ needs to accurately predict the location indicator value $l^P_i$, given the feature-values of $P$ and estimates of all the other location indicator values $\hat{l}^P_j$ (where $j \neq i$). That is, each classifier $C_i$ in the collection assumes that the estimates of the location-indicator values, $\hat{l}^P_j$ for all other locations $s_j$ (where $j \neq i$) are already known, and is responsible for predicting only the indicator value $l^P_i$ for location $s_i$, given all the other indicator values. For a Bayesian network classifier this means calculating the conditional probability

$$\Pr(l^P_i = 1 \mid P, \hat{l}^P_1, \ldots, \hat{l}^P_{i-1}, \hat{l}^P_{i+1}, \ldots, \hat{l}^P_q),$$

under classifier $C_i$, where $\hat{l}^P_1, \ldots, \hat{l}^P_{i-1}, \hat{l}^P_{i+1}, \ldots, \hat{l}^P_q$ are all estimated using simple SVM classifiers. The classifiers $C_1, \ldots, C_q$ are each learned by directly optimizing
an objective function that is based on such conditional probabilities, calculated with respect to the training data as explained in Section 3.1.

The procedures used for learning the Bayesian network classifiers and to combine the individual network predictions are described throughout the rest of the section.

### 3.1 Structure and Parameter Learning of Bayesian Network Classifiers

Given a dataset \( D \), consisting of a set of \( m \) proteins \( \{P_1, \ldots, P_m\} \) and their respective location vectors \( \{l_{P_1}, \ldots, l_{P_m}\} \), each classifier \( C_i \) is trained so as to produce the “best” prediction possible for the value of the location indicator \( l_{P_j}^i \) (for location \( s_i \)), for any given protein \( P \) and a set of estimates of location indicators for all other locations (as shown in Equation 3 above). Based on this aim and on the available training data, we use the Conditional Log Likelihood (CLL) as the objective function to be optimized when learning each classifier \( C_i \). Classifiers whose structures were learnt by optimizing this objective function were found to perform better than classifiers that used other structures [31].

This objective function is defined as:

\[
CLL(C_i \mid D) = \sum_{j=1}^{m} \log Pr(L_i = l_{P_j}^i \mid f_{P_j}, \hat{l}_{P_{j-1}}^i, \ldots, \hat{l}_{P_{j+1}}^i, \ldots, \hat{l}_{P_q}^i).
\]

Each \( P_j \) is a protein in the training set and each probability term is the conditional probability of protein \( P_j \) to have the indicator value \( l_{P_j}^i \) (for location \( s_i \)), given its feature vector \( f_{P_j} \) and the current estimates for all the other location indicators are \( \hat{l}_{P_k}^i \) (where \( k \neq i \)), under the Bayesian network structure \( G_i \) for the classifier \( C_i \) that governs the joint distribution of all the variables in the network (see Equation 2).

To learn a Bayesian network classifier that optimizes this objective function, we use a greedy hill climbing search (see [31,41] for details). While Grossman and Domingos [31] propose a heuristic method that modifies the basic search depicted by Heckerman et al. [41], we do not employ it in this preliminary study, but rather use the basic search, as it does not prove to be prohibitively time consuming. To find estimates for the location indicator values \( \hat{l}_{P_j}^i \), we compute a one-time estimate for each indicator \( l_{P_j}^i \) from the feature-values of the protein \( f_{P_j} \) by using an SVM classifier (e.g. [30]). We use the SVM implementation provided by the Scikit-learn library [42] with a Radial Basis Function kernel. We employ \( q \) such SVMs, \( SVM_1, \ldots, SVM_q \), where each SVM classifier is trained to distinguish one location indicator from the rest, as done in the Binary Relevance approach [43]. The rest of the network parameters are estimated as follows: For each Bayesian network classifier \( C_i \), we use the maximum likelihood estimates calculated from frequency counts in the training dataset, \( D \), to estimate the network parameters (see [31]). To avoid overfitting of the parameters, we apply standard smoothing by adding pseudo-counts for all the events that have zero counts (see [44] for details).
To summarize, at the end of the learning process we have \( q \) Bayesian network classifiers, \( C_1, \ldots, C_q \), like the ones depicted in Figure 1, and \( q \) SVMs, \( SVM_1, \ldots, SVM_q \), used for obtaining initial estimates for each location variable for any given protein. We next describe how these classifiers are used to predict the multi-location of a protein \( P \).

### 3.2 Multiple Location Prediction

Given a protein \( P \), whose locations we would like to predict, we first use the SVMs to obtain preliminary estimates for each of its location indicator values \( \hat{l}_{P1}, \ldots, \hat{l}_{Pq} \). We then use each of the learned classifiers \( C_i \), and the preliminary values obtained from the SVMs to predict the value of the location indicator \( l_i^P \). The classifier outputs a value of either a 0 or a 1 by thresholding, as shown in Equation 1. The conditional probability of \( l_i^P \) given the feature-values of the protein \( P \) and the estimates of the location indicator values \( \hat{l}_{Pj} \) (where \( j \neq i \)) is first calculated as:

\[
Pr(l_i^P = 1 \mid f_P, \hat{l}_{P1}, \ldots, \hat{l}_{Pi-1}, \hat{l}_{i+1}, \ldots, \hat{l}_{Pq}) = \frac{Pr(l_i^P = 1, f_P, \hat{l}_{P1}, \ldots, \hat{l}_{Pi-1}, \hat{l}_{i+1}, \ldots, \hat{l}_{Pq})}{\sum_{z \in \{0,1\}} Pr(l_i^P = z, f_P, \hat{l}_{P1}, \ldots, \hat{l}_{Pi-1}, \hat{l}_{i+1}, \ldots, \hat{l}_{Pq})}.
\]  

(4)

The joint probabilities in the numerator and the denominator of Equation 4 above are factorized into conditional probabilities using the Bayesian network structure, \( G_i \) (see Equation 2). The 0/1 prediction for each \( l_i^P \) obtained from each \( C_i \) becomes the value of the \( i \)’th position in the location-indicator vector \( \langle l_{P1}, \ldots, l_{Pq} \rangle \) for protein \( P \). This is the total multi-location prediction for protein \( P \).

In the next section, we describe our experiments using the Bayesian network framework for predicting protein multi-location and the results obtained.

### 4 Experiments and Results

We implemented our algorithms for learning and using a collection of Bayesian network classifiers as described above using Python and the machine learning library Scikit-learn [42]. We have applied it to a dataset containing single- and multi-localized proteins, previously used for training YLoc+ [29]. Below we describe the dataset, the experiments, the evaluation methods we use, and the multiple location prediction results obtained on the proteins from this dataset.

#### 4.1 Data Preparation

In our experiments we use a dataset containing 5447 single localized proteins (originally published as the Höglund dataset [32]) and 3056 multi-localized proteins (originally published as part of the DBMLoc set [13] that is no longer...
publicly available). The combined dataset was previously used by Briestmeister et al. [29] in their extensive comparison of multi-localization prediction systems. We report results obtained over the multi-localized proteins for comparing our system to other published systems, since the results for these systems are only available for this subset [29]. For all other experiments described here, we report results obtained over the combined set of single- and multi-localized proteins. We use the exact same representation of a 30-dimensional feature vector as used in YLoc+ [29]. The features include sequence-based features, e.g. amino acid composition and those based on PROSITE patterns, as well as on GO annotations. (See [29] for details on the pre-processing, feature construction, and feature selection). The single localized proteins are from the following locations (abbreviations and number of proteins per location is given in parentheses): cytoplasm (cyt, 1411 proteins); endoplasmic reticulum (ER, 198), extra cellular space (ex, 843), golgi apparatus (gol, 150), lysosomal (lys, 103), mitochondrion (mi, 510), nucleus (nuc, 837), membrane (mem, 1238), and peroxisomal (per, 157). The multi-localized proteins are from the following pairs of locations: cyt_nuc (1882 proteins), ex_mem (334), cyt_mem (252), cyt_mi (240), nuc_mi (120), ER_ex (115), and ex_nuc (113). Note that all the multi-location subsets used have over 100 representative proteins.

4.2 Experimental Setting and Performance Measures

To compare the performance of our system to that of other systems (YLoc+ [29], Euk-nPLoc [45], WoLF PSORT [46], and KnowPred [27]), whose performance on a large set of multi-localized proteins was described in a previously published comprehensive study [29], we use the exact same dataset, employing the commonly used stratified 5-fold cross-validation. As the information about the exact 5-way splits used before is not available, we ran five complete runs of 5-fold-cross-validation (i.e. 25 runs in total), where each complete run of 5-fold cross-validation uses a different 5-way split. The use of multiple runs with different splits helps validate the stability and the significance of the results. To ensure that the results obtained by using our 5-way splits for cross-validation can be fairly compared with those reported before [29], we replicated the YLoc+ runs using our 5-way splits, and obtained results that closely match those originally reported by Briestmeister et al. [29]. (The replicated F1-label score is 0.69 with standard deviation of ±0.01, compared to YLoc+ reported F1-label score of 0.68, and the replicated accuracy is 0.65 with standard deviation of ±0.01, compared to YLoc+ reported accuracy of 0.64). The total training time for our system is about 11 hours (wall-clock), when running on a standard Dell Poweredge machine with 32 AMD Opteron 6276 processors. Notably, no optimization or heuristics for improving run time were employed, as this is a one-time training. For the experiments described here, we ran 25 training experiments, through 5 times 5-fold cross validation, where the total run time was about 75 hours (wall clock).

We use in our evaluation the adapted measures of accuracy and F1 score proposed by Tsoumakas [43] for evaluating multi-label classification. Some of these
measures have also been previously used for multi-location evaluation [28,29]. To formally define these measures, let \( D \) be a dataset containing \( m \) proteins. For a given a protein \( P \), let \( M_P = \{ s_i \mid l_P = 1, \text{ where } 1 \leq i \leq q \} \) be the set of locations to which protein \( P \) localizes, and let \( \hat{M}_P = \{ s_i \mid \hat{l}_P = 1, \text{ where } 1 \leq i \leq q \} \) be the set of locations that a classifier predicts for protein \( P \), where \( \hat{l}_P \) is the 0/1 prediction obtained (as described in Section 3). The multi-label accuracy and the multi-label \( F_1 \) score are defined as:

\[
\text{Acc} = \frac{1}{m} \sum_{j=1}^{m} \frac{|M_j \cap \hat{M}_j|}{|M_j \cup \hat{M}_j|} \quad \text{and} \quad \text{F}_1 = \frac{1}{m} \sum_{j=1}^{m} \frac{2|M_j \cap \hat{M}_j|}{|M_j| + |\hat{M}_j|}.
\]

Adapted measures of Precision and Recall, denoted \( \text{Pre}_{s_i} \) and \( \text{Rec}_{s_i} \), are used to evaluate how well our system classifies proteins as localized or not localized to any single location \( s_i \) [29]. The Multilabel-Precision is:

\[
\text{Pre}_{s_i} = \frac{1}{|\{ P \in D \mid s_i \in M_P \}|} \sum_{P \in D \mid s_i \in M_P} \frac{|M_P \cap \hat{M}_P|}{|M_P|}.
\]

and the Multilabel-Recall is:

\[
\text{Rec}_{s_i} = \frac{1}{|\{ P \in D \mid s_i \in M_P \}|} \sum_{P \in D \mid s_i \in M_P} \frac{|M_P \cap \hat{M}_P|}{|\hat{M}_P|}.
\]

Note that \( \text{Pre}_{s_i} \) captures the ratio of the number of correctly predicted multiple locations to the total number of multiple locations predicted, and \( \text{Rec}_{s_i} \) captures the ratio of the number of correctly predicted multiple locations to the number of original multiple locations, for all the proteins that co-localize to location \( s_i \). Therefore, high values of these measures for proteins that co-localize to the location \( s_i \) indicate that the sets of predicted locations that include location \( s_i \) are predicted correctly. Additionally, the \( F_1\)-label score used by Briesemeister et al. [29] to evaluate the performance of multi-location predictors is computed as follows:

\[
\text{F}_1\text{-label} = \frac{1}{|S|} \sum_{s_i \in S} \frac{2 \times \text{Pre}_{s_i} \times \text{Rec}_{s_i}}{\text{Pre}_{s_i} + \text{Rec}_{s_i}}.
\]

Finally, to evaluate the correctness of predictions made for each location \( s_i \), we use the standard precision and recall measures, denoted by \( \text{Pre-Std}_{s_i} \) and \( \text{Rec-Std}_{s_i} \) (e.g. [10]) and defined as:

\[
\text{Pre-Std}_{s_i} = \frac{TP}{TP + FP} \quad \text{and} \quad \text{Rec-Std}_{s_i} = \frac{TP}{TP + FN},
\]

where \( TP \) (true positives) denotes the number of proteins that localize to \( s_i \) and are predicted to localize to \( s_i \), \( FP \) (false positives) denotes the number of proteins that do not localize to \( s_i \) but are predicted to localize to \( s_i \), and \( FN \) (false negatives) denotes the number of proteins that localize to \( s_i \) but are not predicted to localize to \( s_i \).
Table 1. Multi-location prediction results, averaged over 25 runs of 5-fold cross-validation, for multi-localized proteins only, using our system, YLoc⁺ [29], Euk-mPLoc [15], WoLF PSORT [16], and KnowPred [27]. The $F_1$-label score and Acc measures shown for all the systems except for ours are taken directly from Table 3 in the paper by Briesemeister et al. [29]. Standard deviations are provided for our system (not available for other systems).

<table>
<thead>
<tr>
<th></th>
<th>Our system</th>
<th>YLoc⁺ [29]</th>
<th>Euk-mPLoc</th>
<th>WoLF PSORT</th>
<th>KnowPred [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$-label</td>
<td>0.66 (± 0.02)</td>
<td>0.68</td>
<td>0.44</td>
<td>0.53</td>
<td>0.66</td>
</tr>
<tr>
<td>Acc</td>
<td>0.63 (± 0.01)</td>
<td>0.64</td>
<td>0.41</td>
<td>0.43</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Table 2. Multi-location prediction results, averaged over 25 runs of 5-fold cross-validation, for the combined set of single- and multi-localized proteins, using our system. The table shows the $F_1$ score, the $F_1$-label score, and the accuracy (Acc) obtained for SVMs without using location inter-dependencies and for our system which uses location inter-dependencies. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th></th>
<th>SVMs (without using dependencies)</th>
<th>Our system (using dependencies)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
<td>0.77 (± 0.01)</td>
<td>0.81 (± 0.01)</td>
</tr>
<tr>
<td>$F_1$-label</td>
<td>0.67 (± 0.02)</td>
<td>0.76 (± 0.02)</td>
</tr>
<tr>
<td>Acc</td>
<td>0.72 (± 0.01)</td>
<td>0.76 (± 0.01)</td>
</tr>
</tbody>
</table>

Table 3. Multi-location prediction results, per location, averaged over 25 runs of 5-fold cross-validation, for the combined set of single- and multi-localized proteins. Results are shown for the five locations $s_i$ that have the largest number of associated proteins (the number of proteins per location is given in parenthesis): cytoplasm (cyt), extracellular space (ex), nucleus (nuc), membrane (mem), and mitochondrion (mi). The table shows the measures (standard precision [Pre-Std$^a_i$] and recall [Rec-Std$^a_i$], and Multilabel-Precision [Pre$^a_i$] and Multilabel-Recall [Rec$^a_i$]), obtained for SVMs without using location inter-dependencies and for our system by using location inter-dependencies. The highest values between the two methods are shown in boldface. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th>Location</th>
<th>SVMs (without using dependencies)</th>
<th>Our system (without using dependencies)</th>
<th>Our system (using dependencies)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyt (3785)</td>
<td>0.84 (± 0.01)</td>
<td>0.87 (± 0.02)</td>
<td>0.79 (± 0.02)</td>
</tr>
<tr>
<td>ex (1405)</td>
<td>0.91 (± 0.02)</td>
<td>0.79 (± 0.03)</td>
<td>0.90 (± 0.01)</td>
</tr>
<tr>
<td>nuc (2952)</td>
<td>0.90 (± 0.01)</td>
<td>0.90 (± 0.02)</td>
<td>0.90 (± 0.01)</td>
</tr>
<tr>
<td>mem (1824)</td>
<td>0.89 (± 0.01)</td>
<td>0.90 (± 0.02)</td>
<td>0.90 (± 0.01)</td>
</tr>
<tr>
<td>mi (870)</td>
<td>0.87 (± 0.01)</td>
<td>0.90 (± 0.02)</td>
<td>0.89 (± 0.02)</td>
</tr>
</tbody>
</table>

4.3 Classification Results

Table 1 shows the $F_1$-label score and the accuracy for our system in comparison to those obtained by other predictors (as reported by Briesemeister et al. [29], Table 3 there, using the same set of multi-localized proteins and evaluation measures. While the table shows that our system has a slightly lower performance than YLoc⁺, the differences in the values are not statistically significant, and the overall performance level is comparable. Thus our approach performs as effectively as current top-systems, while having the advantage of directly capturing...
inter-dependencies among locations in a generalizable manner (that is, without introducing a new location-class for each new location-combination).

Table 2 shows the $F_1$ score, the $F_1$-label score, and the accuracy obtained by the individual SVM classifiers (used for computing estimates of location indicators) without using location inter-dependencies compared with the corresponding values obtained by our system by using location inter-dependencies, on the combined dataset of both single- and multi-localized proteins. All the scores obtained by using inter-dependencies are significantly higher than those obtained by using SVMs alone without utilizing inter-dependencies. These differences are highly statistically significant ($p < 0.001$), as measured using the 2-sample t-test.

Table 3 shows the prediction results obtained by our system for the five locations that have the largest number of associated proteins: cytoplasm (cyt), extracellular space (ex), nucleus (nu), membrane (mem), and mi (mitochondrion), on the combined dataset of both single- and multi-localized proteins. For each location $s_i$, we show the standard precision ($Pre_{std_i}$) and recall ($Rec_{std_i}$) as well as the Multilabel-Precision ($Pre_{s_i}$) and Multilabel-Recall ($Rec_{s_i}$). The table shows values for each of the measures obtained by SVMs without using location inter-dependencies and by our system using location inter-dependencies. When using inter-dependencies, we note that for all locations the Multilabel-Recall ($Rec_{s_i}$) increases (in some cases statistically significantly); while for a few locations (such as cytoplasm and membrane) the Multilabel-Precision ($Pre_{s_i}$) decreases, the decrease is not statistically significant. For instance, when classifying using SVMs without using inter-dependencies $Rec_{cyt}$ is 0.78 and $Rec_{mem}$ is 0.76, while when incorporating the inter-dependencies the recall is 0.80 and 0.78, respectively. Even for locations with fewer associated proteins, e.g. peroxisome, (157 proteins), the Multilabel-Recall increases from 0.37 using simple SVMs to 0.65 using our classifier. This demonstrates the advantage of using location inter-dependencies for predicting protein locations, not just for locations that have a large number of associated proteins but also for locations that have relatively few associated proteins.

5 Discussion and Future Work

We presented a new way to use a collection of Bayesian network classifiers taking advantage of location inter-dependencies to provide a generalizable method for predicting possible multiple locations of proteins. The results demonstrate that the performance of our preliminary system is comparable to the best current multi-location predictor YLoc+ [29], which indirectly addresses dependencies by creating a class for each multi-location combination. Our results also show that utilizing inter-dependencies significantly improves the performance of the location prediction system, with respect to SVM classifiers that do not use any inter-dependencies.

In most biological applications that have used Bayesian networks so far (e.g. [30, 37, 38]), the variable-space typically corresponds to genes or SNPs which is
a very large space and necessitates the use of strong simplifying assumptions and many heuristics. In contrast, we note that predicting multiple locations for proteins involves a significantly smaller number of variables (as the number of subcellular components and the number of features for representing proteins are relatively small), making this task ideally suitable for the use of Bayesian networks.

The study presented here is a first investigation into the benefit of directly modeling and using location inter-dependencies. In order to obtain initial estimates for location values, we used a simple SVM classifier, and location inter-dependencies were only learned based on these values. While the results have already shown much improvement with respect to the baseline SVM classifiers, we believe that a better approach would be to simultaneously learn a Bayesian network while estimating the location values using methods such as expectation maximization.

We note that although the dataset we use contains the most extensive available collection of multi-localized proteins, several subcellular locations are not represented in the dataset at all due to the low number of proteins associated with them. Similarly, there is not enough data pertaining to proteins that are localized to more than two locations. We are in the process of constructing a set of multi-localized proteins that will be used in future work to test the performance of our system on novel, and more complex, combinations. We also plan to develop improved approaches for learning models of location inter-dependencies from the available data.

Acknowledgments: We are grateful to S. Briesemeister for so readily providing us with information about the implementation and testing of YLoc+

References

27. Lin, H., Chen, C., Sung, T., Ho, S., and Hsu, W.: Protein subcellular localization prediction of eukaryotes using a knowledge-based approach. BMC Bioinformatics, 10:8, 2009