A PROBABILISTIC FRAMEWORK FOR PROTEIN MULTI-LOCATION PREDICTION, AND ITS APPLICABILITY TO MULTI-LABEL CLASSIFICATION

by

Ramanuja Simha

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Summer 2016

A PROBABILISTIC FRAMEWORK FOR PROTEIN MULTI-LOCATION PREDICTION, AND ITS APPLICABILITY TO MULTI-LABEL CLASSIFICATION

by

Ramanuja Simha

Approved: _____

Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____

Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____

Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Hagit Shatkay, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Li Liao, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Robert F. Murphy, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Vijay Shanker, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Cathy H. Wu, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I am extremely grateful to my advisor, Professor Hagit Shatkay, for her support, guidance, and motivation throughout my doctoral study. I cannot thank her enough for the meticulous feedback she has consistently given me over the years. Her deep insights have constantly pushed me beyond my abilities. Without her support, this work would not have been possible.

I am thankful to my committee members Professors Li Liao, Robert Murphy, Vijay Shanker, and Cathy Wu for their thoughtful comments. I am also grateful to Professor Murphy for allowing me to work on a short-term project in his lab at Carnegie Mellon University at the beginning of my study as part of my introduction to interdisciplinary work in biology.

Working in the Computational Biomedicine lab has been invigorating. I would like to extend my gratitude to all the members of the lab, and in particular, Kaidi Ma, Pengyuan Li, Moumita Bhattacharya, Xiangying Jiang, Debarati Roychowdhury, and Gongbo Zhang for providing a fun environment that is also conducive to learning.

My doctoral work could not have been completed without the help of my collaborators. I am thankful to Sebastian Briesemeister and Professor Oliver Kohlbacher for providing access to their protein feature extraction code. I would like to thank Melody Lugo for assisting me in the development of a protein database, and Corey Shannon and Joshua Simmons for helping me develop a web interface for my project.

Finally, without the sacrifices of my parents, I would never have been able to pursue my current career path. I am extremely grateful for their unconditional support throughout my personal journey. I am also thankful to my brother, Raghava Simha, and my sister, Surabhi Simha, for being available whenever I needed.

DEDICATION

To my parents, Sudha Narasimha and Narasimha Rangachar

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

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. While it has been shown that proteins localize to multiple locations, most computational methods assign a single location per protein. A few recent systems assign multiple locations to proteins. However, they typically treat locations as independent of each other.

We present a system for protein multi-location prediction that utilizes inter-dependencies among locations for predicting multiple locations of proteins. Results obtained by using this system show that incorporating such inter-dependencies in the location prediction process improves the classifier's prediction performance.

In machine learning terms, assigning multiple locations to proteins is a special case of multi-label classification (MLC), where proteins can be viewed as instances and locations as instance labels. Improving on the initial system, we introduce an advanced approach for MLC based on a probabilistic generative model that explicitly captures dependencies between features and subsets of labels, in addition to representing inter-dependencies among labels as done by our earlier system. Experimental results demonstrate improved performance of our system for protein multi-location prediction as well as for the general problem of MLC.

The most comprehensive current set of multi-localized proteins used to assess the performance of multi-location prediction systems contains proteins localizing to only *two* locations. We present a procedure to construct a more extensive collection of proteins that localize to multiple locations. This procedure comprises extracting reliable protein information from up-to-date online repositories and storing relevant attributes in a relational database.

Chapter 1

INTRODUCTION

The subcellular location of a protein is important for inferring its biological function and its role within the cell. While much progress has been made during the last decade in the development of computational methods for predicting locations of proteins, most such methods assign a single location per protein. However, research has shown that proteins localize to multiple locations. In this thesis, we present methods that are able to assign possibly multiple locations to proteins.

In machine-learning terms, assigning multiple locations to proteins is a *multi-label classification (MLC)* task. Simple MLC systems assume that labels are independent of one another, while more complex approaches capture label inter-dependencies. Experiments comparing performance of MLC systems demonstrate that there is much room for improvement. We show in this thesis that the method presented here can be used to effectively address the more general problem of MLC.

In Section 1.1, we briefly discuss advantages and shortcomings of existing location prediction systems. Section 1.2 examines current approaches for multi-label classification. In Section 1.3, we present the thesis contributions. Finally, Section 1.4 provides an outline of this thesis.

1.1 Protein Multi-Location

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 [3]. Experimental methods for protein localization such as those based on mass spectrometry [25] or green fluorescence detection [42], although often used in practice, are time consuming and typically not cost-effective for high-throughput localization.

Hence, much ongoing effort has been put into developing high-throughput computational methods [8, 27, 66, 78, 86] to obtain proteome-wide location predictions. Such methods are fast, and can potentially predict locations of proteins whose actual locations have not yet been experimentally determined.

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 by an (over-)simplifying assumption that proteins localize to a single location. However, proteins do localize to multiple compartments within the cell [32, 63, 65, 121], and translocate from one location to another [72]. 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 [76, 80]. As another example, the enzyme *TREX1*, which assists in DNA repair, is primarily present in the cytoplasm but is also transported to the nucleus in response to DNA damage [102]. Thus, predicting multiple locations of proteins is important, since protein movement across locations enables the protein to serve multiple distinct functions.

While a few recent systems (e.g. [10, 14, 17, 56]) attempt to predict multiple locations of proteins, their performance leaves much room for improvement. These systems typically either treat locations as independent and do not attempt to utilize possible inter-dependencies among locations, or represent dependencies only specific to location combinations present in the training set. Since proteins localize systematically, and translocation occurs only among specific locations for the purpose of a particular subcellular function, our hypothesis is that modeling inter-depedencies among locations can assist in predicting locations of proteins more accurately.

1.2 Multi-Label Classification

Traditional single-label classification [104] assigns a single *label* to each *instance*, and is addressed by methods such as Support Vector Machines (e.g. [41, 83]) or naïve

Bayes (e.g. [81]). Multi-label classification, on the other hand, aims to associate each instance with possibly multiple labels.

In the context of machine learning, assigning multiple locations to proteins is a special case of *multi-label classification (MLC)*, where proteins can be viewed as *instances* and locations as *instance labels*. The general problem of MLC is concerned with all classification tasks that assign multiple labels to instances.

Some of the simplest and most commonly used approaches transform the original multi-label prediction task into one or more single-label prediction task(s) (e.g. [105]). However, such approaches typically assume that labels are independent of one another.

More sophisticated approaches for MLC capture inter-dependencies among labels. For example, some methods use multiple binary classifiers, one classifier per label. To capture label inter-dependencies, the *feature-vector* used to represent an input instance given to a classifier includes label assignments obtained from other classifiers (e.g. [77]). Similar systems that explicitly learn the inter-dependencies employ Bayesian networks (e.g. [1, 119, 120]). Other approaches include systems that utilize probabilistic generative models (e.g. [62, 79, 108]), typically built for classifying text. These systems have not been extensively tested on datasets other than text.

However, current MLC systems do not capture intricate dependencies between features and labels, and experiments demonstrate that their performance can still be much improved. We hypothesize that by explicitly modeling dependencies between features and subsets of labels using sophisticated techniques, the assignment of multi-labels to instances can be done more accurately. Moreover, thus far, the advanced methods for MLC that represent label inter-dependencies (e.g. [1, 119]) have not been employed by any of the work on predicting locations of proteins.

1.3 Thesis Contributions

The primary goal of this research is to develop new methods for protein multi-location that capture location inter-dependencies. We apply some of these methods to a number of more general multi-label classification tasks such as predicting emotions of

songs and predicting labels of scenes. We thus demonstrate that the resulting impact of our work extends well beyond protein multi-location prediction. Performance of all methods presented in this thesis have been shown to be either comparable to or better than that of state-of-the-art systems. Our research contributions are as follows.

1. A protein multi-location prediction system that achieves improved performance by using location inter-dependencies [89, 90]. We present a system that utilizes inter-dependencies among locations for predicting multiple locations of proteins. The system comprises a collection of Bayesian network classifiers. Each classifier is used to assign a single location to a protein. Notably, this system does not capture intricate dependencies and independencies between features and locations. We evaluate our system on a comprehensive set of multi-localized proteins derived from the DBMLoc dataset, previously used to assess the performance of multi-location prediction systems. Results obtained by using our system show that utilizing location inter-dependencies improves the prediction performance.

2. A multi-label classification system that improves upon the performance of state-of-the-art prediction systems [91, 92, 94, 95]. Improving on our initial system, we introduce a prediction method for the general problem of multi-label classification (MLC). This method utilizes a probabilistic generative model that explicitly captures dependencies between features and subsets of labels, in addition to representing label inter-dependencies as done by our initial system. We evaluate this approach on the same dataset of multi-localized proteins used above to assess the performance of our initial system. The results obtained by the advanced method improve upon those obtained by our initial simpler classifier as well as by other top systems. To demonstrate the wide applicability of the generative model based approach in the general context of MLC, we applied the related system to a number of MLC tasks using standard multi-label datasets. Prediction results showed that the performance of our system significantly improves upon results obtained by other current MLC systems reported in a previously published comprehensive study.

3. An extensive collection of multi-localized proteins [93]. The most comprehensive available current set of multi-localized proteins, derived from DBM-Loc [121], consists of proteins that localize to only *two* locations. We present a procedure to construct a database that contains proteins localizing to more than two locations. This procedure constitutes first extracting reliable protein information from up-to-date online data sources, and then developing a relational database to store the information.

1.4 Thesis Overview

In Chapter 2, we review current approaches for protein multi-location prediction and for multi-label classification. Chapter 3 presents an initial system for protein multi-location prediction that employs a collection of Bayesian network classifiers. In Chapter 4, we introduce a sophisticated approach for the general problem of multi-label classification that is based on a probabilistic generative model. Chapter 5 discusses results obtained by the new method on a dataset of multi-localized proteins as well as other standard multi-label datasets. Chapter 6 details a procedure to construct an extensive collection of multi-localized proteins from online repositories. In Chapter 7, we present thesis summary and outline directions for future work.

Chapter 2

RELATED WORK

We survey here research methods for protein multi-location prediction and for multi-label classification. In Section 2.1, we provide details of the machine learning techniques used in systems that predict multiple locations of proteins, and discuss their advantages and shortcomings. Protein multi-location prediction is a special case of the more general problem of multi-label classification, as pointed out earlier in Chapter 1. In Section 2.2, we thus also review approaches for multi-label classification, which is concerned with all classification tasks that predict multiple labels of instances.

2.1 Methods for Protein Multi-Location Prediction

A number of recent location prediction systems attempt to predict multiple locations of proteins, however their performance leaves much room for improvement. While most use sequence-derived features (e.g., amino acid composition) and Gene Ontology terms to represent proteins, and employ machine learning methods to predict protein locations, a few are based solely on sequence-based similarity. The former class of methods incorporate one or more of the following classifiers: k-nearest neighbors (k-NN) [17]), Support Vector Machines (SVMs) [56], and naïve Bayes [10].

For instance, ngLOC [54] uses a naïve Bayes classifier to obtain a probability distribution over locations for each query protein, where the location probabilities are computed *independently* of each other. For the two most probable locations, a *confidence score* that reflects the likelihood of the protein localizing to both locations is computed. If the score is above a certain threshold, both locations are assigned to the protein. This method is limited to assigning at most two locations to proteins.

Several methods use variations of k-NN to assign multiple locations to proteins. For example, a system introduced by Li et al. [56] utilizes multiple binary classifiers, where each classifier comprises an ensemble of k-NN and SVMs that distinguishes between a pair of locations and provides a vote for one of the locations. The location with the highest number of votes is assigned to a query protein. If multiple locations have the same majority vote, a multi-location prediction is made, in which all the locations with the majority vote are assigned to the protein.

As another example, WoLF PSORT [46, 47] uses a k-NN classifier that employs a distance measure based on a weighted sum of the Euclidean and Manhattan distances. This system assigns a query protein to the location combination that is most common among the protein's k nearest neighbors, thus limiting the method to only assigning multi-locations present in the training set. Furthermore, the performance of WoLF PSORT was evaluated using an extensive dataset [10] and was found to be significantly lower than the performances of top performing systems (see e.g. [10, 57]).

Other k-NN-based methods include similar systems introduced by the same group such as Euk-mPLoc [14] and iLoc-Euk [17]. These two systems both compute a score for each location, based on the query protein. iLoc-Euk assigns the protein to the locations with the highest scores; the number of locations assigned is the same as that associated with the nearest neighbor protein in the dataset. Euk-mPLoc assigns the query protein to locations whose scores lie within a certain deviation from the highest score. iLoc-Euk utilizes relevant GO terms as a part of its protein representation, and to achieve the reported level of performance, the system strongly relies on features derived from these terms — which are only available for proteins that are *already annotated*. Moreover, iLoc-Euk was not extensively tested against current multi-location predictors, and the performance of Euk-mPLoc was found to be significantly lower than that of current systems (see e.g. [10, 57]).

A number of methods similar to iLoc-Euk were proposed by the same group for localizing subsets of eukaryotic proteins [18, 114], virus proteins [117], and bacterial proteins [115, 116]. Additional domain-specific systems similar to Euk-mPLoc have

been introduced as well (Euk-mPLoc 2.0 [15], Hum-mPLoc 2.0 [87], Plant-mPLoc [16], and Virus-mPLoc [88]).

In contrast to all the above approaches that employ machine learning methods, KnowPred$_{\text{site}}$ [57] uses similarity between protein sequences to predict multiple locations of proteins. The collection is built by extracting for each protein in the training dataset, peptide fragments from its sequence and from similar sequences. Each fragment is annotated with the locations associated with the original protein. The peptide fragments for a query protein are obtained in a similar manner, and the system uses the location annotations of matching peptide fragments in the collection to compute a score for each location. Using the two highest location scores, a *confidence score* is computed to determine if the protein should be assigned multiple locations. This method is restricted to assigning at most two locations to a protein (similar to that seen earlier for ngLOC [54]).

Notably, none of the above methods incorporate inter-dependencies among locations into the location prediction process. All the methods assign locations to proteins without accounting for the influence of one location assignment on another.

Recent work by He et al. [43] attempts to take advantage of *correlations* among locations when assigning multiple locations to proteins. As part of the training process, their classifier (namely, Imbalanced Multi-modal Multi-label Learning) attempts to learn a correlation measure between pairs of locations, which is later used to make location assignments. While this system takes into account a simple type of dependency among locations, namely, pair-wise correlations between locations, it does not account for more complex inter-dependencies. Furthermore, the system was not tested on any extensive protein multi-localization dataset.

YLoc$^+$[10] introduces a new class for each combination of locations supported by the training set, and thus captures only dependencies specific to these location combinations. The system uses a naïve Bayes classifier to obtain a probability distribution over location combinations for each query protein. The initial distribution is then transformed into a multinomial distribution over the individual locations. YLoc$^+$

reports the locations whose probabilities are higher than some threshold as the multi-location prediction for a query protein. We note that as the number of possible location combinations is exponential in the number of locations. Furthermore, training the naïve Bayes classifier over only location combinations in the training set does not provide a practical model in the general case of multi-localized proteins that localize to other multi-locations. While using the most comprehensive set of multi-localized proteins, the prediction results obtained by YLoc$^+$ were found to be the best among those obtained by current multi-location prediction systems [10].

All the above systems for protein multi-location prediction address a special case of multi-label classification. We next review approaches for the more general problem.

2.2 Approaches for Multi-Label Classification

In the context of machine learning, assigning multiple locations to proteins is a special case of multi-label classification (MLC), where proteins can be viewed as *instances* and locations as *instance labels*. The general problem of MLC is concerned with all classification tasks that assign multiple labels to instances.

Several basic approaches for MLC transform the original prediction task into one or more single-label classification task(s). For instance, under the Label Powerset method [105], each unique combination of labels in the training set is considered as a class, and the prediction task constitutes assigning each input instance to one of the classes. This method's assignments are restricted to label combinations within the training set, and the classifier learned does not provide a practical prediction model in the general case of multi-label instances beyond the training set. Moreover, since the number of classes can be exponential in the number of labels, classifier learning can be computationally expensive.

Other similar methods include the Ranking by Pairwise Comparison method [105] and the Binary Relevance method [105]. For the former method, a classifier is trained to distinguish between each possible pair of labels (one-vs-one). In the case of the latter, each classification task corresponds to distinguishing a single label from the rest

(one-vs-all). Thus, Binary Relevance employs a classifier for each label and is a computationally efficient alternative to Ranking by Pairwise Comparison, which utilizes an exponential number of classifiers.

The number of instances associated with some classes/labels may be significantly lower than that associated with others. Thus, in the case of the above approaches, transforming an MLC task into multiple single-label classification tasks can result in the poor performance of individual classifiers responsible for predicting such classes/labels.

More advanced approaches for MLC typically employ a variant of BR, and capture dependencies among labels. For example, a *classifier chain* [77] consists of multiple binary classifiers like those used in Binary Relevance, one classifier per label, and the chain is constructed based on an input label ordering. To capture relationships among labels, the *feature-vector* used to represent an input instance given to a classifier F includes label assignments obtained from all classifiers preceding classifier F in the chain. A multi-label assignment for the instance is determined by combining the predictions obtained from all binary classifiers. A probabilistic variant of Classifier Chains [22] cannot be practically applied to multilabel datasets that contain more than a few labels, as it is computationally complex.

Systems based on Classifier Chains that explicitly *learn* label inter-dependencies include a chain of Support Vector Machines [120], a chain of naïve Bayes classifiers [119], and an ensemble of Bayesian networks [1]. The ordering used to construct a chain of classifiers is fixed using a topological ordering of labels in a Bayesian network.

Among the rest of the approaches for MLC are systems that utilize probabilistic generative models (see e.g. [62, 79, 108]), typically built for classifying text. These systems have not been extensively tested against other multi-label classification systems and on datasets other than text.

Notably, when an instance is associated with multiple labels, a feature-value of the instance may depend only on a subset of these labels. Current systems do not model the dependence of features on label subsets. Moreover, performance of these systems leaves much room for improvement.

In the rest of this thesis, we present approaches for protein multi-location prediction as well as for the more general problem of multi-label classification. We first present an initial approach for predicting multiple locations of proteins that captures location inter-dependencies. Improving on the initial method, we introduce a new approach for multi-label classification. This new approach represents intricate dependencies between instances and labels, in addition to capturing label inter-dependencies. Employing our approaches to address a number of multi-label classfication tasks, we present experiments and results over the most extensive dataset of multi-localized proteins currently publicly available as well as over several multi-label datasets that were previously used to assess the performance of multi-label classification systems.

Chapter 3

A DEPENDENCY-BASED SYSTEM FOR PROTEIN (MULTI-)LOCATION PREDICTION

We present here an initial system that we introduced for predicting locations of multiply-localized proteins. While proteins move from one location to another and can localize to multiple locations, we pointed out in the introductory chapter that most existing location prediction systems assign only a single location per protein. Our extensive survey of multi-location prediction methods (Chapter 2) points out that a few recent systems do attempt to assign multiple locations to proteins. However, such systems typically treat locations as independent and do not attempt to utilize possible inter-dependencies among locations.

In this chapter, we present a system based on a collection of Bayesian network classifiers — which we denote by *BNCs* — that incorporates label inter-dependencies into the process of assigning locations to proteins. The Bayesian network related to each classifier corresponds to a single location. Learning the structure of each Bayesian network classifier takes into account inter-dependencies among locations, and the prediction process utilizes estimates involving multiple locations. We employ BNCs to predict locations of proteins by utilizing a comprehensive dataset of single and multi-localized proteins, and compare the performance of our system with that of current multi-location prediction systems.

In Section 3.1, we formulate protein multi-location prediction as classification via Bayesian networks, and present a brief background on Bayesian networks along with the relevant notations. Section 3.2 discusses the procedures used for learning the network structure and parameters, and for assigning multiple locations to proteins. Section 3.3 provides details about the dataset used in our evaluation, the performance

evaluation measures, and the experimental results. Finally, Section 7.1 summarizes our findings and outlines possible directions for improving our system.

3.1 Problem Formulation

As is commonly done in the context of classification, and protein location prediction in particular [10, 35, 45], we represent each protein, P, as a weighted feature vector, $\vec{f^P}=\langle f_1^P, \ldots, f_d^P \rangle$, where d is the number of features. Each feature represents a characteristic of a protein, such as the presence or absence of a short amino acid motif [27, 45], the relative abundance of a certain amino acid as part of amino-acid composition [54], or the annotation by a Gene Ontology (GO) term [48]. We view each feature vector entry f_j^P as a value taken by a random variable F_j with respect to protein P.

To represent a protein's locations, let $S=\{s_1, \ldots, s_q\}$ be the set of q possible subcellular components in the cell. Each protein P localizes to at least one — and possibly more than one — location. The locations of P are represented by a location indicator vector, $\vec{l^P}=\langle l_1^P, \ldots, l_q^P \rangle$ of 0/1 values, where $l_i^P=1$ if P localizes to s_i, and $l_i^P=0$ otherwise. As with the feature values, each location vector entry, l_i^P, is viewed as a value taken by a random variable L_i. Given a dataset consisting of m proteins along with their location vectors, we denote the dataset as: $D=\{(P_j, \vec{l^{P_j}}) \mid 1 \leq j \leq m\}$. 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 represented as a feature-vector, $\vec{f^P}$, assigns a (correct) 0/1 value to each of the entries l_i^P making up P's location-indicator vector, $\vec{l^P}$.

Consider a subset of subcellular locations $s_{i_1}, \ldots, s_{i_k}$. Recall that we use the random variables L_i to denote whether a protein is localized or not to location s_i. Formally, the locations in the set, $s_{i_1}, \ldots, s_{i_k}$, are considered *independent* of each other if for any protein P, the joint probability of P to be in any of these locations can be

written as the product of the individual location probabilities, that is:

$$Pr(L_{i_1}=l_{i_1}^P,\ldots,L_{i_k}=l_{i_k}^P)=\prod_{j=1}^{k}\Pr(L_{i_j}=l_{i_j}^P)\ .$$

If the locations are *not independent*, that is, if for a protein P,

$$Pr(L_{i_1}=l_{i_1}^P,\ldots,L_{i_k}=l_{i_k}^P)\neq\prod_{j=1}^{k}\Pr(L_{i_j}=l_{i_j}^P)\ ,$$

then we say that these locations are *inter-dependent*.

Our underlying hypothesis, which is supported by the experiments and the results presented in this chapter, is that utilizing such location inter-dependencies can form the basis for an improved approach for location-prediction. Bayesian networks have been used before in many biological applications [33, 55, 85]. We use them here to model inter-dependencies among subcellular locations, as well as among protein-features and locations.

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 assigning the 0/1 value, l_i^P, indicating P's non-localization or localization to s_i. To calculate location assignments for a protein P, each classifier C_i utilizes inter-dependencies among estimates of location values of P, $\hat{l}_j^P$ (for all other locations s_j, where $j\neq i$), along with the feature-values of P. We use support vector machines (SVMs) [41, 83] to compute these estimates. The output of classifier C_i for a protein P is given by

$$C_i(P)=\begin{cases}1 & \text{If } \Pr(l_i^P=1\mid P,\hat{l}_1^P,\ldots,\hat{l}_{i-1}^P,\hat{l}_{i+1}^P,\ldots,\hat{l}_q^P)>0.5\ ;\\ 0 & \text{Otherwise .}\end{cases}\tag{3.1}$$

We briefly introduce Bayesian networks here, along with the relevant notations; see [50] for more details. A Bayesian network consists of a directed acyclic graph G, whose nodes are random variables, which in our case represent features, denoted $F_1,\ldots,F_d$, and locations, 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 presented by Fayyad and Irani [29] and used by Dougherty et al. [24] to discretize the features; details about the discretization technique are presented in Section 3.3.1. This technique was also used in the development of YLoc$^+$ [10].

Directed edges in the graph indicate inter-dependencies among the random variables. As demonstrated in Figure 3.1, 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. While this assumption may oversimplify the underlying biological mechanisms, it works well in practice and has proven useful before [10], and it helps speed up the process of learning the network structure from the data. Further details about the learning procedure itself are provided later in Section 3.2.

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, θ_v, containing the conditional probabilities of the values the node takes given its parents' values, $\Pr(v \mid Pa(v))$. We denote by Θ the set of all conditional probability tables, and the Bayesian network is the pair (G, Θ). A consequence of using the Bayesian network structure is that it represents certain *conditional independencies* among non-neighboring nodes [50], such that the joint distribution of the set of network variables can be simply calculated as:

$$\Pr(F_1, \ldots, F_d, L_1, \ldots, L_q) = \prod_{i=1}^{d} \Pr(F_i \mid Pa(F_i)) \prod_{j=1}^{q} \Pr(L_j \mid Pa(L_j)) \; . \tag{3.2}$$

Figure 3.1 shows an example of a collection of Bayesian network classifiers, $C_1, \ldots, C_q$, one for each of the q subcellular locations $s_1, \ldots, s_q$. Each classifier C_i consists of the graph G_i and its set of parameters Θ_i, (Θ_i not shown in the figure). The task of classifier C_i is to infer the value of the location variable L_i. As such, L_i is viewed as *unobserved*, and is shown as an unshaded node in the figure. The values

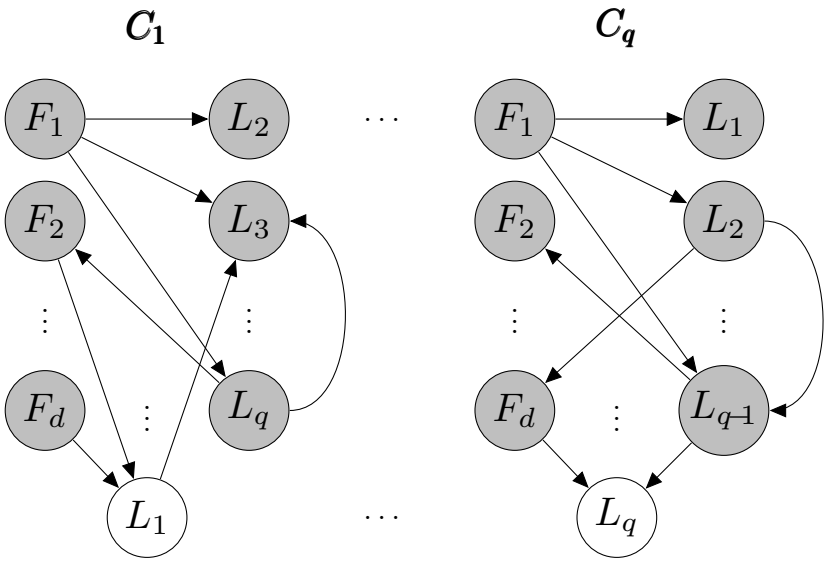


Figure 3.1: An example of a collection of Bayesian network classifiers. The collection consists of several classifiers $C_1, \ldots, C_q$, one for each of the q subcellular locations. Nodes represent random variables. Directed edges represent dependencies between the variables. There are edges among 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.

of feature variables $F_1, \ldots, F_d$ are given for each protein. Thus, these variables are viewed as known or *observed*, and are shown as shaded nodes. Finally, the values of the location variables for all locations except for L_i, $\{L_1, \ldots, L_q\} - \{L_i\}$, are needed for inferring the value of L_i in the classifier C_i. As such, these variables are viewed by the classifier as though they are *observed*. Notably, the values of these variables are not known and therefore need to be estimated.

The structure and parameters of the network for each classifier C_i are learned as described in the next section. The task of each classifier C_i, is to infer 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 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. We note that here we set out to show that capturing inter-dependencies among locations help improve prediction, and the relatively simple estimation procedure that we use serves sufficiently well. Other methods, such as expectation maximization, can be used to estimate all the hidden parameters, which we shall do in methods introduced in the next chapter.

3.2 Model Learning and Inference

As our goal is to assign (possibly multiple) locations to proteins, we use a collection of Bayesian network classifiers, where each classifier C_i, infers the value (0 or 1) of a single location variable L_i – while using *estimates* of values taken by all other location variables L_j $(j \neq i)$. The location estimates are assumed to be known as far as classifier C_i is concerned and are calculated using SVM classifiers as described in Section 3.2.1. The location inferences 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 the training data before it can be used. As described in Section 3.1, each classifier C_i consists of a graph structure G_i and a set of conditional probability parameters, Θ_i; i.e., $C_i = (G_i, \Theta_i)$. 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 assign locations to proteins.

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

$$\Pr(l_i^P = 1 \mid P, \hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P) \ , \tag{3.3}$$

where $\hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P$ 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 3.2.1.

The procedures used for learning the Bayesian network classifiers and for combining individual location inferences are described throughout the rest of this section.

3.2.1 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 $\{\vec{l}^{P_1}, \ldots, \vec{l}^{P_m}\}$, each classifier C_i is trained so as to produce the "best" inference possible for the location indicator value, l_i^P (for location s_i), for any given protein P and estimates of all the other location indicator values (as shown in Equation 3.3 above). Based on this aim and on the available training data, we use *Conditional Log Likelihood (CLL)* as the objective function to be optimized when learning each classifier C_i. Classifiers whose structures were learned by optimizing this objective function have been found to achieve improved prediction performance [38]. The objective function is thus defined as:

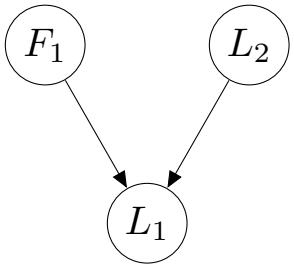
$$CLL(C_i \mid D) = \sum_{j=1}^{m} \log \Pr(L_i = l_i^{P_j} \mid \vec{f}^{P_j}, \hat{l}_1^{P_j}, \ldots, \hat{l}_{i-1}^{P_j}, \hat{l}_{i+1}^{P_j}, \ldots, \hat{l}_q^{P_j}) \ .$$

Each P_j is a protein in the training set, and each probability term in the sum is the conditional probability of protein P_j to have the indicator value $l_i^{P_j}$ (for location s_i), given its feature vector $\vec{f}^{P_j}$ and the current estimates of all the other location indicator values $\hat{l}_k^{P_j}$ (where $k \neq i$), under the Bayesian network structure G_i for the classifier C_i (see Equation 3.2).

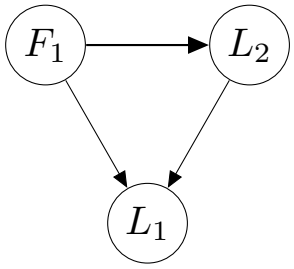
To learn a Bayesian network classifier that optimizes the above objective function, we use a greedy hill climbing search. While Grossman and Domingos [38] proposed a heuristic method that modifies the basic search depicted by Heckerman et al. [44], we do not employ it for learning structures, but rather use the basic search, as the latter does not prove to be prohibitively time consuming. Our structure learner starts with an initial network with no directed edges. In each iteration of the hill climbing algorithm, a directed edge is either added, deleted, or its direction reversed. An example of each of the possible steps is shown in Figure 3.2. Notably, we do not allow the introduction of directed edges that connect two feature variables to one another. This constraint accounts for the assumption incorporated into the network structure, as discussed in Section 3.1, of independence or conditional independence among the features given the locations; it slightly simplifies the network structure and reduces the search space and the overall learning time.

To find estimates $\hat{l}_k^{P_j}$ of location indicator values, we compute a one-time estimate for each indicator $l_i^{P_j}$ from the feature values of the protein $\vec{f}^{P_j}$ by using an SVM classifier (e.g. [41, 83]). We employ q SVM classifiers, $SVM_1, \ldots, SVM_q$, where each SVM classifier, SVM_i is trained to distinguish a single location s_i from the rest. We use the SVM implementation provided by the Scikit-learn library [68] with a Radial Basis Function kernel.

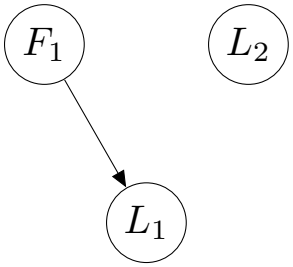
The rest of the network parameters are estimated as follows. For each Bayesian network classifier C_i, we use the maximum likelihood estimates calculated from frequency counts in the training dataset, D, to estimate the network parameters. For each node v in the graph G_i, (where v may either be a feature variable or a location variable), we denote its n parents as $Pa(v) = \{Pa_1(v), \ldots, Pa_n(v)\}$. For each value x of



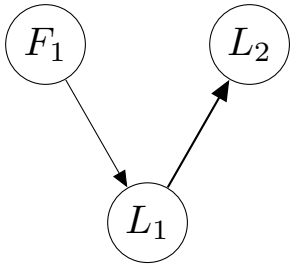
(i)



(ii) Adding an edge (F_1, L_2).



(iii) Deleting an edge (L_2, L_1).



(iv) Reversing an edge (L_2, L_1) to (L_1, L_2).

Figure 3.2: Adding, deleting, and reversing an edge in a Bayesian network during structure learning. The network on the top left (i), is the starting point. Networks (ii), (iii), and (iv) show the addition, deletion, and reversal of an edge, respectively, as performed by the greedy hill climbing algorithm for structure learning.

v and values $y_1, \ldots, y_n$ of its respective parents, the conditional probability parameter $\Pr(v{=}x \mid Pa_1(v){=}y1, \ldots, Pa_n(v){=}y_n)$ is computed as follows: Let n_{joint} be the number of proteins in the dataset D for whom the value of variable v is x and the values of $Pa_1(v), \ldots, Pa_n(v)$ are $y_1, \ldots, y_n$, respectively. Let $n_{marginal}$ be the number of proteins in the dataset D whose values of the variables denoted by $Pa_1(v), \ldots, Pa_n(v)$ are $y_1, \ldots, y_n$ (regardless of the value of variable v). The maximum likelihood estimate for the conditional probability is thus:

$$\Pr(v = x \mid Pa_1(v) = y_1, \ldots, Pa_n(v) = y_n) = \frac{n_{joint}}{n_{marginal}} \ .$$

To avoid overfitting of the parameters, we add pseudo-counts to events that have zero counts (a variation on Laplace smoothing [60]).

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 3.1 (one for each of the q locations), and q SVM classifiers, $SVM_1, \ldots, SVM_q$, used for obtaining initial estimates of values taken by each location variable for any given protein.

We next describe how these classifiers are used to obatain a multi-location prediction for a protein P.

3.2.2 Multiple Location Prediction

Given a protein P, whose locations we would like to infer, we first use the SVMs to obtain estimates, $\hat{l}_1^P, \ldots, \hat{l}_q^P$, of location indicator values. We then use the learned Bayesian network classifier C_i and the location estimates obtained from the SVMs to infer the location indicator value, l_i^P. The classifier outputs a value of either a 0 or a 1, as shown in Equation 3.5. The entire process is depicted in Figure 3.3. The conditional probability of l_i^P given the feature values of the protein P and the estimates of the location indicator values $\hat{l}_j^P$ (where $j \neq i$) is calculated as:

$$\Pr(l_i^P = 1 \mid \vec{f^P}, \hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P) = \frac{\Pr(l_i^P = 1, \vec{f^P}, \hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P)}{\sum_{z \in \{0,1\}} \Pr(l_i^P = z, \vec{f^P}, \hat{l}_1^P, \ldots, \hat{l}_{i-1}^P, \hat{l}_{i+1}^P, \ldots, \hat{l}_q^P)} \ . \tag{3.4}$$

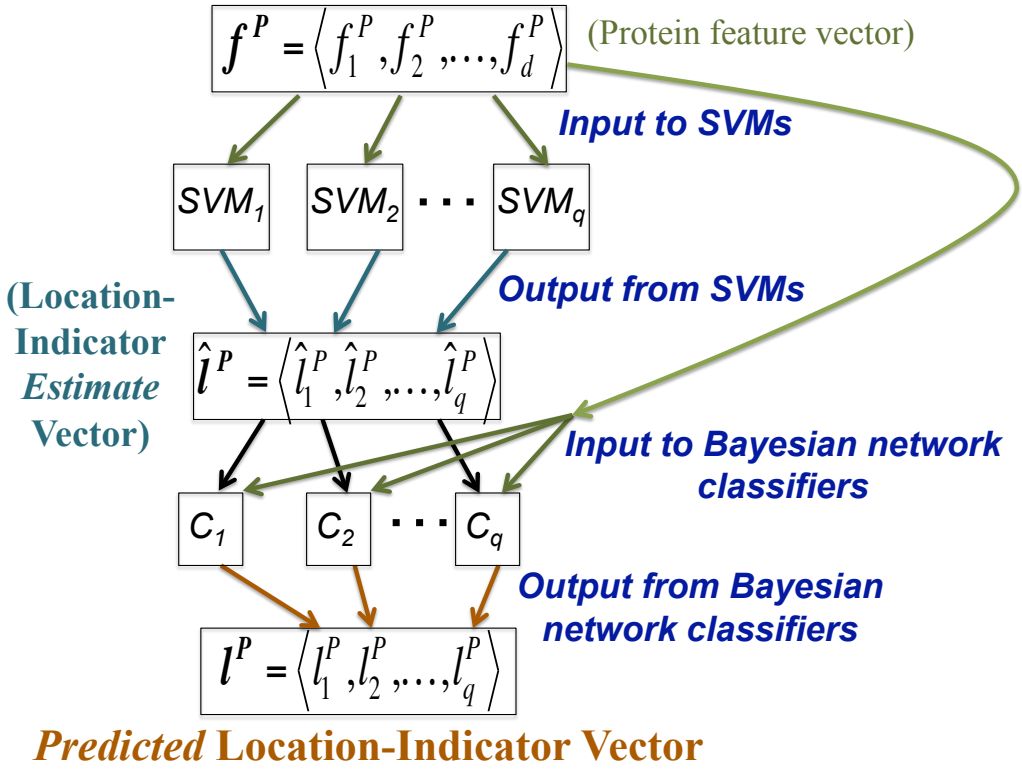


Figure 3.3: Procedure used to assign multiple locations to a protein P. First, SVM classifiers, $SVM_1, \ldots, SVM_q$, are used to obtain location estimates, $\hat{l}_1^P, \ldots, \hat{l}_q^P$. The Bayesian network classifiers, $C_1, \ldots, C_q$ are then used to utilize the inter-dependencies among the location estimates and infer actual location values, $l_1^P, \ldots, l_q^P$.

The joint probabilities in the numerator and the denominator of Equation 3.4 above are factorized into conditional probabilities using the Bayesian network structure, G_i (see Equation 3.2). 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 $\langle l_1^P, \ldots, l_q^P \rangle$ for protein P. This is the complete multi-location prediction for protein P.

In the next section, we describe our experiments using the Bayesian network framework and the results obtained.

3.3 Experiments and Results

We implemented our algorithms for learning the Bayesian network classifiers and for assigning multi-locations to proteins using Python and the machine learning library Scikit-learn [68]. We applied our system to predict locations of proteins using a dataset containing single- and multi-localized proteins, previously used for training YLoc$^+$ [10]. We describe the dataset in Section 3.3.1, present experiments and evaluation measures in Section 3.3.1, and discuss multi-location prediction results in Section 3.3.2.

3.3.1 Data

In our experiments we use a dataset containing 5447 single-localized proteins (originally published as part of the Höglund dataset [45]), and a collection of 3056 multi-localized proteins (originally published as part of the DBMLoc set [121]). The combined dataset was constructed and previously used by Briesemeister et al. [10] in their extensive comparison of multi-localization prediction systems. Notably, the protein sequences from the Höglund dataset share less than or equal to 30% sequence identity with each other, while sequences from the DBMLoc dataset share less than 80% sequence similarity with each other.

We report results obtained over the *3056 multi-localized proteins* for comparing our system to other published systems, since the results for these systems are only available for this subset [10]. For all other experiments described here, we report results obtained over the combined set of single- and multi-localized proteins. The

single-localized proteins are from the following locations (abbreviations and number of proteins per location are given in parentheses): cytoplasm (*cyt*, 1411 proteins); endoplasmic reticulum (*ER*, 198), extra cellular space (*ex*, 843), golgi apparatus (*gol*, 150), lysosome (*lys*, 103), mitochondrion (*mi*, 510), nucleus (*nuc*, 837), membrane (*mem*, 1238), and peroxisome (*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.

Protein Representation. We use the exact same representation of a 30-dimensional feature vector as used by Briesemeister et al. for YLoc$^+$ [10, 11], described below. However, we also run experiments in which we do not use annotation-based features (items iii and iv in the list below) in the protein representation.

(i) Thirteen features derived directly from the protein sequence data, specifically, length of the amino acid chain, length of the longest very hydrophobic region, respective number of Methionine, Asparagine, and Tryptophane, occurring in the N-terminus, number of small amino acids occurring in the N-terminus, and numerical values based on: (a) ER retention signal, (b) peroxisomal targeting signal, (c) clusters of consecutive Leucines occurring in the N-terminus, (d) secretory pathway sorting signal, (e) putative mitochondrial sorting signal;

(ii) Nine features contructed using pseudo-amino acid composition [13], which are based on certain physical and chemical properties of amino acid subsequences;

(iii) Two *annotation-based* features constructed using two distinct groups of PROSITE patterns, one characteristic of plasma-membrane proteins and the other of nucleus proteins. For each protein, the value of the respective feature is 1 if the protein sequence contains at least one PROSITE pattern characteristic of the organelle, 0 otherwise;

(iv) Six *annotation-based* features based on GO-annotations. Five of these correspond to five location-specific GO terms [GO:0005783 (endoplasmic reticulum), GO:0005739 (mitochondrion), GO:0005576 (extracellular region), GO:0042025 (host

cell nucleus), and GO:0005778 (peroxisomal membrane)], where the feature value is 1 if at least one sequence homologous to the protein's is associated with the GO term according to Swiss-Prot (release 42.0), 0 otherwise. The sixth feature indicates the likely location of the protein given all the GO terms assigned to it (or to its homologues) in Swiss-Prot.

(See Briesemeister et al. [10, 11] for further details regarding the pre-processing, feature construction, and feature selection.)

Feature Discretization. To ensure that all feature values are discrete, we use the minimal entropy partitioning technique as initially presented by Fayyad and Irani [29] and used by Dougherty et al. [24]. We rephrase the partitioning technique by using concepts from Information Theory, in particular, the definition of *conditional entropy* [19]. Each continuous-valued feature is converted into a discrete-valued feature by recursively dividing the range of values that the feature obtains into intervals; all feature values lying within an interval are mapped to a single discrete feature value.

Formally, for a training set of m proteins associated with q locations $s_1, \ldots, s_q$, we denote the range of values assigned to feature f_i for proteins in the set by $[l_{f_i}, h_{f_i}]$, where l_{f_i} is the lowest value in the range and h_{f_i} the highest. A *discretization boundary* T_i partitions the feature value range $[l_{f_i}, h_{f_i}]$ into two intervals, $[l_{f_i}, T_i]$ and $(T_i, h_{f_i}]$. Given a boundary T_i, for each protein P_j in the set (where $1 \leq j \leq m$), its feature value for feature f_i, denoted f_i^j, is mapped to a value $d_1^{T_i}$ if $f_i^j \in [l_{f_i}, T_i]$ and to another value $d_2^{T_i}$ if $f_i^j \in (T_i, h_{f_i}]$, where $d_1^{T_i}$ and $d_2^{T_i}$ are two distinct values, chosen from the set $\{0, 1, 2, \ldots\}$ (e.g. $d_1^{T_i} = 0$ and $d_2^{T_i} = 1$).

Each location s_k ($1 \leq k \leq q$), with which a protein P_j may be associated, is viewed as a value taken by a random variable S. The conditional probability distribution of S given a value f_i^j for feature f_j and the discretization boundary T_i is defined as:

$$\Pr(S|f_i^j, T_i) = \begin{cases} \Pr(S|f_i^j \leq T_i) & \text{if } f_i^j \leq T_i \ ; \\ \Pr(S|f_i^j > T_i) & \text{if } f_i^j > T_i \ . \end{cases} \tag{3.5}$$

The respective conditional entropy is denoted $H(S|f_i^j, T_i)$ [19] and defined as:

$$H(S|f_i^j, T_i) = -\Pr(f_i^j \leq T_i) \sum_{k=1}^{q} \Pr(S = s_k|f_i^j \leq T_i) \log_2(\Pr(S = s_k|f_i^j \leq T_i)) \\ -\Pr(f_i^j > T_i) \sum_{k=1}^{q} \Pr(S = s_k|f_i^j > T_i) \log_2(\Pr(S = s_k|f_i^j > T_i)) \ ,$$

where $\Pr(f_i^j \leq T_i)$ is estimated as the proportion of proteins in the training set whose feature value for f_i is less than or equal to T_i, $\Pr(f_i^j > T_i)$ is estimated as the proportion of proteins whose feature value for f_i is greater than T_i, $\Pr(s_k|f_i^j \leq T_i)$ is estimated by the proportion of proteins associated with location s_k among those whose feature value for f_i is less than or equal to T_i, and $\Pr(s_k|f_i^j > T_i)$ is estimated by the proportion associated with s_k among those proteins whose feature value for f_i is greater than T_i. The discretization boundary T_i is chosen such that the conditional entropy $H(S|f_i^j, T_i)$ is minimal.

The partitioning into intervals is applied recursively, and terminates when a stopping condition based on the *Minimum Description Length Principle*, (see Fayyad and Irani [29] for details), is satisfied. This recursive partitioning is independently applied to each of the features.

Exclusion of Annotation-based Features. It has been shown by several research groups [8, 48, 106] that location prediction performance is improved by incorporating features based on GO-annotations associated with each protein (which may also include location annotation) into the protein representation. However, we note that an important goal of protein location prediction is to assign locations to proteins that are *not yet annotated*; that is, the location prediction tool may serve as an aid in the protein annotation process. Therefore, it is useful to be able to accurately predict location of proteins even without using annotation-based features such as PROSITE patterns and GO terms.

To test the performance of our system with and without such features, we have constructed several versions of the dataset in which we include/exclude PROSITE-based and GO-based features. (i) *PROSITE-GO* — which includes both PROSITE-

and GO-based features in the protein representation; (ii) *No-PROSITE-GO* — which does not include any PROSITE- or GO-based features in the protein representation; (iii) *No-PROSITE* — which does not include PROSITE-based features, but *includes* GO-based features; and (iv) *No-GO* — which does not include any GO-based features, but *includes* PROSITE-based features, in the protein representation. These datasets are used as described in Section 3.3.2 to demonstrate that location inter-dependencies can be used to improve prediction performance, even in the absence of PROSITE-based and GO-based features.

To compare the performance of our system to that of other systems (YLoc$^+$ [10], Euk-mPLoc [14], WoLF PSORT [47], and KnowPred$_{\text{site}}$ [57]), whose performance on a large set of multi-localized proteins was described in a previously published comprehensive study [10], 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 in previous studies 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 statistical 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 [10], we replicated the YLoc$^+$ runs using our 5-way splits, and obtained results that closely match those originally reported by Briestmeister et al [10]. (The replicated F_1-*label* score is 0.69 with standard deviation ± 0.01, compared to YLoc$^+$ reported F_1-*label* score of 0.68, and the replicated accuracy is 0.65 with standard deviation ± 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 in this chapter, 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 F_1 *score* proposed by Tsoumakas et al. [105] for evaluating multi-label classification. Some of these measures have also been previously used to evaluate multi-location prediction systems [10, 43]. To formally define these measures, let D be a dataset containing m proteins. For a given protein P, let $M^P = \{s_i \mid l_i^P = 1, \text{ where } 1 \leq i \leq q\}$ be the set of locations to which protein P localizes according to the dataset, and let $\hat{M}^P = \{s_i \mid \hat{l}_i^P = 1, \text{ where } 1 \leq i \leq q\}$ be the set of locations that a classifier predicts for protein P, where $\hat{l}_i^P$ is the 0/1 prediction obtained (as described in Section 3.2). The multi-label accuracy and the multi-label F_1 score are defined as:

$$ML_{acc} = \frac{1}{|D|} \times \sum_{P \in D} \frac{|M^P \cap \hat{M}^P|}{|M^P \cup \hat{M}^P|} \quad \text{and} \quad F_1 = \frac{1}{|D|} \times \sum_{P \in D} \frac{2|M^P \cap \hat{M}^P|}{|M^P| + |\hat{M}^P|} \text{ , respectively.}$$

To evaluate how well our system classifies proteins as localized or not localized to each individual location s_i, we use *adapted* measures of multi-label precision and recall denoted Pre_{s_i} and Rec_{s_i} and defined as follows [10]:

$$Pre_{s_i} = \frac{1}{|\{P \in D | s_i \in \hat{M}^P\}|} \times \sum_{P \in D | s_i \in \hat{M}^P} \frac{|M^P \cap \hat{M}^P|}{|\hat{M}^P|} \text{ ;}$$

$$Rec_{s_i} = \frac{1}{|\{P \in D | s_i \in M^P\}|} \times \sum_{P \in D | s_i \in M^P} \frac{|M^P \cap \hat{M}^P|}{|M^P|} \text{ .}$$

We use here the terms *Multilabel-Precision* and *Multilabel-Recall* to refer to Pre_{s_i} and Rec_{s_i}, respectively. Note that Pre_{s_i} captures the ratio of the number of correctly predicted multi-locations to the total number of multi-locations *predicted*, and Rec_{s_i} captures the ratio of the number of correctly predicted multi-locations to the number of *original* multi-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 combinations of locations that include s_i are predicted correctly.

Additionally, the F_1-*label* score used by Briesemeister et al. [10] to evaluate the performance of multi-location predictors is computed as:

$$F_1\text{-}label = \frac{1}{|S|} \times \sum_{s_i \in S} \frac{2 \times Pre_{s_i} \times Rec_{s_i}}{Pre_{s_i} + Rec_{s_i}} \text{ .}$$

Finally, to evaluate the correctness of predictions made for each location s_i, we use the *standard precision* and *recall* measures, denoted by $Pre\text{-}Std_{s_i}$ and $Rec\text{-}Std_{s_i}$ (e.g. [86]) and defined as:

$$Pre\text{-}Std_{s_i} = \frac{TP}{TP + FP} \quad \text{and} \quad Rec\text{-}Std_{s_i} = \frac{TP}{TP + FN} \ ,$$

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.

3.3.2 Classification Results

Table 3.1 shows the F_1-*label* score and the accuracy of our system based on a collection of Bayesian network classifiers (denoted BNCs) obtained when running over the PROSITE-GO version of the dataset, which includes both PROSITE- and GO-based features in the protein representation, in comparison to those obtained by other multi-location predictors (as reported by Briesemeister et al. [10], *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 (as indicated by the standard deviations of the scores obtained by our system), 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 — that is, without introducing a new location-class for each new location-combination.

Tables 3.2 and 3.3 both show the F_1 score, the F_1-*label* score, and the accuracy obtained by the SVM classifiers compared with the corresponding values obtained by the BNCs, on the *combined dataset of both single- and multi-localized proteins*. The SVMs are used for computing estimates of location indicator values and do not capture location inter-dependencies, whereas the BNCs utilize location inter-dependencies. Table 3.2 displays the scores obtained over the PROSITE-GO version of the dataset.

Table 3.1: Prediction results for *multi-localized* proteins only on the PROSITE-GO version of the dataset, averaged over 25 runs of 5-fold cross-validation. The table shows the F_1-*label* score and the overall accuracy (ML_{acc}) obtained using the BNCs, YLoc$^+$[10], Euk-mPLoc [14], WoLF PSORT [47], and KnowPred$_{\text{site}}$ [57]. All values except ours are taken directly from *Table 3* in the paper by Briesemeister et al. [10]; standard deviations are not available there. The highest values are shown in boldface.

Location Prediction System	$\boldsymbol{F_1}$**-label**	$\boldsymbol{ML_{acc}}$
BNCs	0.66 (± 0.02)	0.63 (± 0.01)
YLoc$^+$	**0.68**	**0.64**
Euk-mPLoc	0.44	0.41
WoLF PSORT	0.53	0.43
KnowPred$_{\text{site}}$	0.66	0.63

Table 3.2: Prediction results for the *combined set* of single- and multi-localized proteins on the PROSITE-GO version of the dataset, averaged over 25 runs of 5-fold cross-validation. The table shows the F_1 score, the F_1-*label* score, and the overall accuracy (ML_{acc}) obtained using SVMs, which do not capture location inter-dependencies and using BNCs, which utilize location inter-dependencies. Standard deviations are shown in parentheses. The highest values are shown in boldface.

	$\boldsymbol{F_1}$	$\boldsymbol{F_1}$-*label*	$\boldsymbol{ML_{acc}}$
SVMs (without using dependencies)	0.77 ($\pm$ 0.01)	0.67 ($\pm$ 0.02)	0.72 ($\pm$ 0.01)
BNCs (using dependencies)	**0.81** ($\pm$ 0.01)	**0.76** ($\pm$ 0.02)	**0.76** ($\pm$ 0.01)

Table 3.3: Prediction results on the No-PROSITE-GO, No-PROSITE, and No-GO versions of the dataset for the same *combined set* of proteins shown in Table 3.2, averaged over 25 runs of 5-fold cross-validation. The table shows the same measures obtained using the same systems shown in Table 3.2. Standard deviations are shown in parentheses. The highest values are shown in boldface.

	Dataset	F_1	F_1-*label*	$MLacc$
SVMs (without using dependencies)	No-PROSITE-GO	0.75 (± 0.04)	0.66 (± 0.02)	0.70 (± 0.04)
BNCs (using dependencies)	No-PROSITE-GO	**0.78** (± 0.05)	**0.72** (± 0.07)	**0.73** (± 0.05)
SVMs (without using dependencies)	No-PROSITE	0.77 (± 0.01)	0.66 (± 0.02)	0.72 (± 0.01)
BNCs (using dependencies)	No-PROSITE	**0.80** (± 0.01)	**0.75** (± 0.02)	**0.75** (± 0.01)
SVMs (without using dependencies)	No-GO	0.76 (± 0.03)	0.67 (± 0.03)	0.71 (± 0.03)
BNCs (using dependencies)	No-GO	**0.79** (± 0.04)	**0.72** (± 0.08)	**0.74** (± 0.04)

In contrast, Table 3.3 displays the scores obtained over dataset versions that do not include the respective annotation-based features in the protein representation, namely, No-PROSITE-GO, No-PROSITE, and No-GO. All the scores in Tables 3.2 and 3.3 obtained using inter-dependencies are higher than those obtained by using SVMs alone that do not utilize inter-dependencies. All differences in Table 3.2 and those corresponding to No-PROSITE in Table 3.3 are statistically significant ($p \ll 0.001$), as measured by the 2-sample t-test [21] when running over the PROSITE-GO, No-PROSITE, and No-GO versions of the dataset.

Table 3.3 shows that location inter-dependencies improve multi-location prediction even when annotation-based features, which utilize PROSITE or GO, are not included in the feature set representing the protein. Furthermore, we see from Tables 3.2 and 3.3 that the performance of the BNCs do not deteriorate substantially when running over dataset versions that do not include various annotation-based features. Thus, BNCs show robustness to the presence/absence of annotation-based features.

Table 5.4 shows per location prediction results obtained by our system when running over the PROSITE-GO version of the dataset for locations that are associated with multi-localized proteins: cytoplasm (cyt), extracellular space (ex), nucleus (nu), membrane (mem), and mi (mitochondrion), on the *combined dataset of both single- and multi-localized proteins*. We don't report results for *endoplasmic reticulum* here as the number of multi-localized proteins associated with this location is relatively very small. For each location s_i, we show *standard precision* ($Pre\text{-}Std_{s_i}$) and *recall* ($Rec\text{-}Std_{s_i}$) as well as *Multilabel-Precision* (Pre_{s_i}) and *Multilabel-Recall* (Rec_{s_i}). The table shows values for each of the measures obtained by the SVMs, which do not capture location inter-dependencies, and by the BNCs, which utilize location inter-dependencies. For a few locations, such as *cytoplasm* and *membrane*, the *Multilabel-Precision* (Pre_{s_i}) decreases when using the inter-dependencies. Moreover, the value of this measure does not improve for locations with relatively a large number of proteins. We note that our system assigns each protein to all locations whose probability exceeds 0.5 and thus to more locations that the protein actually localizes, resulting in a lower precision.

Table 3.4: Per location prediction results for the *combined set* of single- and multi-localized proteins on the PROSITE-GO version of the dataset, averaged over 25 runs of 5-fold cross-validation. 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 *standard precision* ($Pre\text{-}Std_{s_i}$) and *recall* ($Rec\text{-}Std_{s_i}$), as well as *Multilabel-Precision* (Pre_{s_i}) and *Recall* (Rec_{s_i}), for each location s_i. The various per-location scores are obtained using the SVMs and using the BNCs. For each location and measure, the highest of the values obtained from the two methods is shown in boldface. Standard deviations are shown in parentheses.

	cyt (3785)	**ex (1405)**	**nuc (2952)**	**mem (1824)**	**mi (870)**
$\boldsymbol{Pre\text{-}Std_{s_i}}$ **(SVMs)**	**0.84** (± 0.01)	0.87 (± 0.02)	**0.79** (± 0.02)	**0.93** (± 0.01)	**0.90** (± 0.03)
$\boldsymbol{Pre\text{-}Std_{s_i}}$ **(BNCs)**	**0.84** (± 0.01)	**0.91** (± 0.02)	**0.79** (± 0.03)	0.90 (± 0.01)	0.87 (± 0.03)
$\boldsymbol{Rec\text{-}Std_{s_i}}$ **(SVMs)**	0.85 (± 0.01)	0.64 (± 0.02)	0.72 (± 0.02)	0.79 (± 0.02)	0.62 (± 0.03)
$\boldsymbol{Rec\text{-}Std_{s_i}}$ **(BNCs)**	**0.86** (± 0.01)	**0.65** (± 0.02)	**0.74** (± 0.03)	**0.80** (± 0.02)	**0.66** (± 0.03)
$\boldsymbol{Pre_{s_i}}$ **(SVMs)**	**0.82** (± 0.01)	0.89 (± 0.02)	**0.83** (± 0.01)	**0.92** (± 0.01)	0.87 (± 0.03)
$\boldsymbol{Pre_{s_i}}$ **(BNCs)**	0.81 (± 0.02)	**0.91** (± 0.02)	**0.83** (± 0.01)	0.90 (± 0.01)	**0.89** (± 0.02)
$\boldsymbol{Rec_{s_i}}$ **(SVMs)**	0.78 (± 0.01)	0.72 (± 0.02)	0.77 (± 0.01)	0.76 (± 0.01)	0.68 (± 0.02)
$\boldsymbol{Rec_{s_i}}$ **(BNCs)**	**0.80** (± 0.01)	**0.74** (± 0.02)	**0.78** (± 0.02)	**0.78** (± 0.01)	**0.73** (± 0.02)

Nevertheless, most of the differences are not highly statistically significant ($p > 0.01$), as measured by the 2-sample t-test [21]. The *Multilabel-Recall* (Rec_{s_i}) increases for all locations with the use of inter-dependencies where the differences in most cases are highly statistically significant ($p \ll 0.001$).

To demonstrate improved prediction performance of our system across different locations, we examine the statistically-significant differences in the *Multilabel-Recall* for cytoplasm (3785 proteins), membrane (1824), and peroxisome (157). The *Multilabel-Recall* for *cytoplasm* (Rec_{cyt}) increases from 0.78 when classifying by SVMs without using inter-dependencies, to 0.80 when incorporating inter-dependencies. The *Multilabel-Recall* for *membrane* (Rec_{mem}) increases from 0.76 to 0.78 under similar conditions. Even for a location like *peroxisome* that has fewer associated proteins, the *Multilabel-Recall* increases from 0.37 using simple SVMs to 0.65 using our classifier. Our analysis demonstrates the advantage of using location inter-dependencies for inferring protein locations, not just for locations that have a large number of associated proteins but also for locations that are associated with relatively few proteins.

3.4 Summary and Directions for Improvement

We presented a collection of Bayesian network classifiers, taking advantage of location inter-dependencies, to provide a new method for predicting possible multiple locations of proteins. The results demonstrate that the performance of our system is comparable to the current best performing multi-location predictor YLoc$^+$[10]. The latter 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. Moreover, this improved performance due to the use of location inter-dependencies is maintained even when the protein representation does not include PROSITE patterns-based features or GO-based features, thus exhibiting robustness to the presence/absence of annotation-based features.

This study constituted a first investigation into the benefit of directly modeling and using location inter-dependencies. While the location inter-dependencies were only learned based on one-time estimates of location values, the results already show much improvement with respect to the baseline SVM classifiers.

In the next chapter, we present a methodology that learns structure and parameters of the model using iterative techniques. As predicting multiple locations of proteins is a special case of multi-label classification, we present a generative model to address the problem of multi-label classification.

Chapter 4

MULTI-LABEL CLASSIFICATION: USING LABEL INTER-DEPENDENCIES VIA A GENERATIVE MIXTURE MODEL

Improving on the initial location prediction method introduced in the previous chapter, we present here a new approach for multi-label classification. Due to the general applicability of the new approach, we anticipate our work to have impact well beyond protein location prediction, in domains such as scene identification and text classification.

The new method utilizes a probabilistic framework that represents inter-dependencies among labels, and unlike our preliminary method presented in Chapter 3, captures intricate dependencies between features and labels as well. When an instance is associated with multiple labels, a feature-value of the instance may depend only on a subset of these labels and thus be *conditionally independent* of the others given the label-subset. Our initial system as well as current MLC systems do not account for such conditional independence. In this chapter, we present a probabilistic generative model that captures dependencies among labels as well as between features and labels, by means of a Bayesian network. We introduce the concept of *label dependency sets* as a basis for a new mixture model that represents conditional independencies of features from labels given subsets of inter-dependent labels.

In Section 6.1, we provide the motivation for developing a new and improved method for MLC. In Section 4.2, we introduce relevant notations and present our probabilistic generative model. Section 4.3 discusses procedures used for learning structure and parameters of the model, and inference techniques applied for multi-label classification. Finally, Section 5.3 summarizes our contributions and outlines future directions.

4.1 Motivation

Multi-label classification (MLC) associates instances with possibly multiple labels, in contrast to *single-label classification*, where each instance is associated with a single label. While simple approaches for MLC (e.g. [105]) transform the task into one or more single-label classification task(s), more advanced approaches (e.g. [1, 22, 62, 77, 79, 108, 119, 120]) capture dependencies among labels. Notably, in the context of MLC, a feature-value of an instance typically depends on some subset of the instance labels and thus may be *conditionally independent* of the other labels given this subset. For example, the *grade* feature value of college students who are associated with the two labels, *Topped-Exams* and *Secured-Jobs* is typically *High*, regardless of any other labels. Furthermore, dependence of a feature-value on a label is likely to suggest its dependence on other inter-dependent labels. Current systems do not account for such intricate dependencies between feature values and labels. Moreover, performance of current methods leaves much room for improvement. Our hypothesis is that explicitly modeling the dependencies between feature values and *inter-dependent* labels, as part of the classifier model, can support a more accurate assignment of multi-labels to instances.

We thus present a probabilistic generative model that captures dependencies among labels as well as between features and labels, by means of a Bayesian network. We introduce a mixture model to represent conditional independencies of features from labels given subsets of *inter-dependent* labels, and further develop a multi-label classifier. Unlike previous approaches, our system uses an iterative process to infer values for multiple labels simultaneously. In each iteration, the Bayesian network is modified to reflect inter-dependencies among the *most recently inferred label values*; the accuracy of the updated label assignments is thus improved by capturing specific feature-label correlations and dependencies.

In the next section, we describe the model framework, and present our probabilistic generative model.

4.2 A Dependency-Based Mixture Model for Multi-Label Data

Let D be a dataset containing m instances, and $C=\{c_1,\ldots,c_q\}$ be a set of q *class*-labels. Each instance in D is associated with a subset of labels. As others have done before in the context of multi-label classification (MLC) [1, 22], we represent an instance $I \in D$ as a feature vector, $\vec{f}^I=\langle f_1^I,\ldots,f_d^I\rangle$, and I's labels as a label vector, $\vec{l}^I=\langle l_1^I,\ldots,l_q^I\rangle$. Here d is the number of features, and $l_i^I=1$ if instance I is associated with label c_i, $l_i^I=0$ otherwise. Each feature value f_j is viewed as a value taken by a feature random variable F_j, and each label-indicator value l_i is viewed as a value taken by a label random variable L_i. The task of multi-label classification thus amounts to developing a classifier that takes as input an instance represented by a feature vector, and outputs a q-dimensional label vector.

4.2.1 Model Framework

We use a Bayesian network framework to model inter-dependencies among labels as well as between features and labels. Each node represents either a label variable L_i ($1 \leq i \leq q$) or a feature variable F_j ($1 \leq j \leq d$), and each directed edge indicates a dependence relationship between a pair of variables.

Representing label inter-dependencies

In the context of multi-label classification, labels may be directly correlated with one another, regardless of their association with any specific instance. For example, drivers that are labeled as *Speeding* are also likely to be labeled *Accident-Prone*, regardless of any specific driver characteristics (features). We represent each *unconditional dependence* between a pair of labels c_i and c_j as a directed edge from label variable L_i to variable L_j. Figure 4.1 shows an example Bayesian network structure we learn over label variables in the context of the *Emotions* dataset [103], where instances are *songs* and labels are *emotions*. A directed edge in the figure, specifically, from *Amazed-Surprised* to *Sad-Lonely* represents the assertion that knowing that an instance is associated with the label *Amazed-Surprised* influences the level of belief about the instance's association with the label *Sad-Lonely*.

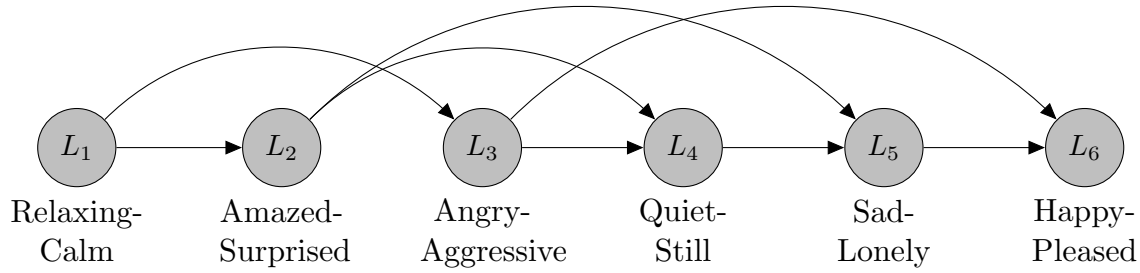


Figure 4.1: An example Bayesian network structure over labels (or emotions) that we learn using the Emotions dataset [103]. Nodes represent random variables; the label associated with each variable is shown below the corresponding node. Directed edges represent dependencies between the variables.

A label may often depend on a small set of a few labels while being conditionally independent of other labels given this set. To continue the previous example, the label *Accident-Prone* is conditionally independent of the label *New-Driver* given the label *Speeding*. To capture such *conditional independencies*, we introduce the concept of *label dependency sets*. A *dependency set* for a label c_i is a minimal set of labels, $c_{i_1}, \ldots, c_{i_m}$ such that knowing an instance's association with each of the labels c_{i_j} in the set is sufficient to infer the likelihood of the instance to be associated with c_i. We utilize the Bayesian network framework to obtain a practical representation of a label dependency set. The network structure captures both direct dependencies between pairs of labels and conditional independencies among certain labels given the others. More specifically, each label variable L_i directly depends on its parents $\mathrm{Pa}(L_i)$ while being conditionally independent of its non-descendants given $\mathrm{Pa}(L_i)$ (for additional details see [81]); the joint distribution of the label variables is thus given by:

$$\Pr(L_1, \ldots, L_q) = \prod_{i=1}^{q} \Pr(L_i | \mathrm{Pa}(L_i)) \ . \tag{4.1}$$

Employing the above conditional independence, we refer to a label variable L_i and its parents in the network as the *label dependency set* for L_i. Thus, for each variable L_i ($1 \leq i \leq q$), we define a label dependency set:

$$LS_i = \{L_i\} \cup \mathrm{Pa}(L_i) \ . \tag{4.2}$$

Representing dependencies between features and labels

An instance's association with certain labels is clearly correlated with the instance feature values. Additionally, the value of a feature may be correlated with multiple labels and not just with one. For example, in the *Emotions* dataset [103] discussed earlier, where instances are songs that are represented using *rhythm* and *tone* features and labels are *emotions*, the value of the tone feature is typically *Low* when a song is labeled as *Sad-Lonely* while it is typically *High* when a song is simultaneously labeled as both *Sad-Lonely* and *Amazed-Surprised*. As another example in a protein database derived from DBMLoc [121], where instances are proteins represented using

amino-acid composition based features and labels are *subcellular locations* of proteins, the number of residues of amino acid *Tryptophan (Trp)*, is typically *High* for a protein that localizes to *Membrane* only; such an abundance agrees with the well-established importance of *Trp*'s role in membrane proteins [82]. In contrast, the number of *Trp* residues for a protein that localizes to both *Membrane* and *Cytoplasm* is much lower.

As explained earlier while introducing label dependency sets (LDS), the association of an instance with certain labels typically provides information about its association with other labels. For instance, a song that has a *High* tone feature value and is labeled as *Amazed-Suprised* is likely to also be labeled as *Sad-Lonely*. We represent the *dependence* of a feature, F_j, on a subset of inter-dependent labels, LS_i (to which we refer as a label dependency set) by plotting directed edges connecting each label variable in the set with the feature variable. We thus capture the *conditional dependence among labels* in the set LS_i given the feature F_j. Figure 4.2 extends the example presented earlier in Figure 4.1. The dashed arrows from each label, L_2, L_4, L_5 (i.e., *Amazed-Surprised*, *Quiet-Still*, and *Sad-Lonely*), to the feature, F_1 (i.e. *Tone*) in the new figure capture the dependence of the feature on the subset of inter-dependent labels comprising the label L_5 and its parents, $L2$ and $L4$; this label subset is referred to as the label dependency set, $LS_5 = \{L_5\} \cup Pa(L_5)$.

Moreover, when an instance is associated with multiple labels, a feature-value of the instance may depend only on a subset of these labels. As an example, the *tone* feature value of a song that is labeled as *Sad-Lonely* and *Amazed-Suprised* is likely to be *High* regardless of the song's association with any other labels. That is, the *tone* is conditionally independent of all other labels, given the two labels *Sad-Lonely* and *Amazed-Suprised*. By explicitly representing *dependence* between a feature and the labels in an LDS as discussed above, our model captures *conditional independence* of the feature from all other labels given the LDS.

We next present a probabilistic generative model that captures the label inter-dependencies and dependencies between features and labels discussed above.

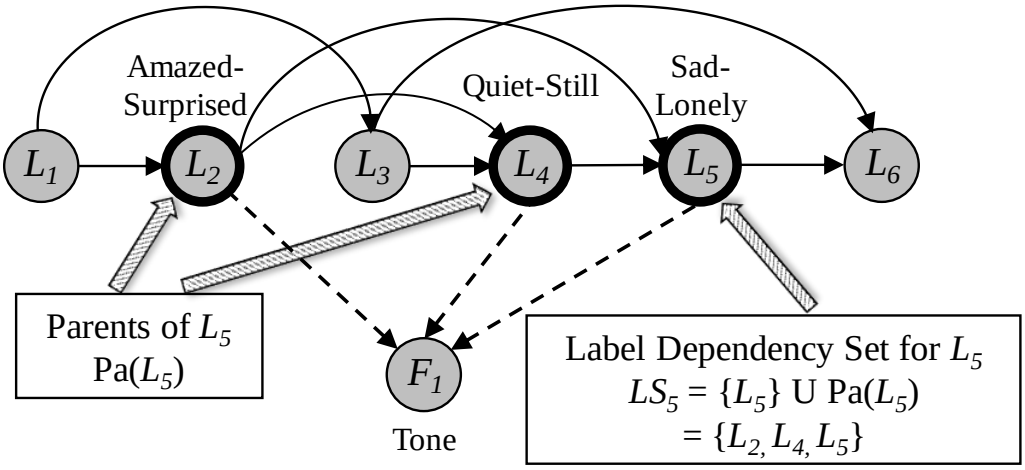


Figure 4.2: An example Bayesian network structure over labels (or emotions) and features (or rhythms/tones) that we learn using the Emotions dataset [103]. (This figure extends the example shown in Figure 4.1.) Nodes represent label/feature random variables. The feature associated with the variable F_1 is shown below the corresponding node. Label variables corresponding to nodes with thick borders (i.e. L_5 and its parents, L_2 and L_4) make up the label dependency set LS_5; the label associated with each variable in the set is shown below the corresponding node. Solid directed edges represent inter-dependencies among the labels. Dashed directed edges capture dependence between the feature, F_1, and the subset of inter-dependent labels in LS_5.

4.2.2 Model Description

Generative models have been used before for multi-label classification (MLC) [62, 79, 108], typically for classifying text. While these models address dependencies among labels, they do not represent intricate dependencies between feature values and subsets of labels. In contrast, our proposed model captures conditional independencies of features from labels by directly representing the dependencies between feature values and label subsets. (In addition, our model is developed in the general context of MLC — not limited to text.) We next discuss the *instance generation process*, based on a Bayesian network structure, and provide further detail about our generative model.

The generation process comprises two steps:

(A) Labels assignment: To generate an instance I, its class-labels are first determined, i.e., a label value l_i^I is assigned to each label variable L_i ($1 \leq i \leq q$). We view each label assignment as a *Bernoulli* event, where l_i^I=1 when I is associated with the label c_i, and l_i^I=0 otherwise. Based on the Bayesian network structure, the probability of L_i to be assigned 1 is denoted as: $\alpha_i = \Pr(L_i{=}1|\text{Pa}(L_i))$ while its probability to be assigned 0 is $1-\alpha_i$. The order in which label values are assigned is based on the topological order of label variables, $L_{t1}, \ldots, L_{tq}$ in the Bayesian network. The assigned label values form a label-vector $\vec{l^I}$ for instance I.

(B) Features assignment: Based on the label-vector $\vec{l}$, a label dependency set (LDS) LS_{F_j} is selected for each feature F_j. We expect this set to constitute a small subset of labels such that F_j's value depends only on the label subset. We introduce a *Multinomial* random variable λ^{F_j} that takes on a value $k \in \{1, \ldots, q\}$ with probability: $\theta_{j,k}^{\vec{l}} = \Pr(\lambda^{F_j}{=}k|\vec{l})$; $\lambda^{F_j}{=}k$ indicates the selection of the k'th LDS. We denote this set as: $LS_k = \{L_k\} \cup Pa(L_k)$. We refer to $\theta_{j,k}^{\vec{l}}$ as the *mixture parameter* as it models the influence of knowing the labels in each LDS LS_k on the belief of the value of feature F_j.

The value for feature F_j is thus selected based on the values taken by the random variables in the set LS_k, denoted as $\mathcal{V}_{LS_k}$. We view each feature value selection as a *Multinomial* event, where the probability of F_j to take on a value v is:

$\phi_{j,k}(v) = \Pr(F_j = v | \lambda^{F_j} = k, \mathcal{V}_{LS_k})$. Here $\phi_{j,k}(v)$ denotes the conditional probability of feature F_j to take on the value v given the label dependency set LS_k. In the model, all feature variables are assumed to take on discrete values. We thus discretize each real-valued feature in the datasets used for our experiments, as done by earlier studies [1] (see Chapter 5 for further details regarding discretization). The selected values for all features together form a complete feature-vector $\vec{f}^I$ representing the instance I.

We view the assignment of label values using a *coin-toss model*, and view the assignment of feature values based on two *die-roll processes*. For each feature, one die roll is used to select an LDS, and another to select a value for the feature. A summary of the generative process for an instance I is shown in Figure 4.3: First, biased *label coins* are tossed in the order $C_{t1}, C_{t2}, \ldots, C_{tq}$ (as shown in the top of *box A* in the figure). If the coin C_{ti} ($1 \leq i \leq q$) comes up *Heads*, the label variable L_{ti} is set to 1 (i.e. $l^I_{ti} = 1$); otherwise L_{ti} is set to 0 (i.e. $l^I_{ti} = 0$). The probability of C_{ti} to come up *Heads* is α_i. Collectively, this results in choosing the label-vector $\vec{l}^I$ (*box A*, bottom).

Next, for each feature F_j ($1 \leq j \leq d$), a biased *label vector die* $D^{F_j}_{\vec{l}^I}$ is rolled; if the die lands with the k'th face up, the LDS LS_k is selected (*box B*, top). The probability of $D^{F_j}_{\vec{l}^I}$ to land with the k'th face up is $\theta^{\vec{l}}_{j,k}$. Based on the selected set LS_k, a biased *feature die* $D^{F_j}_k$ is rolled; if the die lands with the v'th face up, the feature value f^I_j is set to v (*box B*, bottom). The probability of $D^{F_j}_k$ to land with the v'th face up is $\phi_{j,k}(v)$.

We note that the process of assigning feature values enforces several independence relationships. Having selected LS_k as the label dependency set, we denote by $\overline{L}_k$ the set of all label variables other than $\{L_k\} \cup \mathrm{Pa}(L_k)$, i.e., $\overline{L}_k = \{L_1, \ldots, L_q\} - LS_k$. The value selection — for each feature F_j — is *conditionally independent* of all labels in $\overline{L}_k$ given LS_k. Furthermore, the selected feature-value for F_j is also *conditionally independent* of all other feature values given the label vector $\vec{l}^I$.

Formally stated, the *independence assumptions* enforced by our model are:

(i) The feature values $f^I_1, \ldots, f^I_d$ of an instance I, are *conditionally independent*

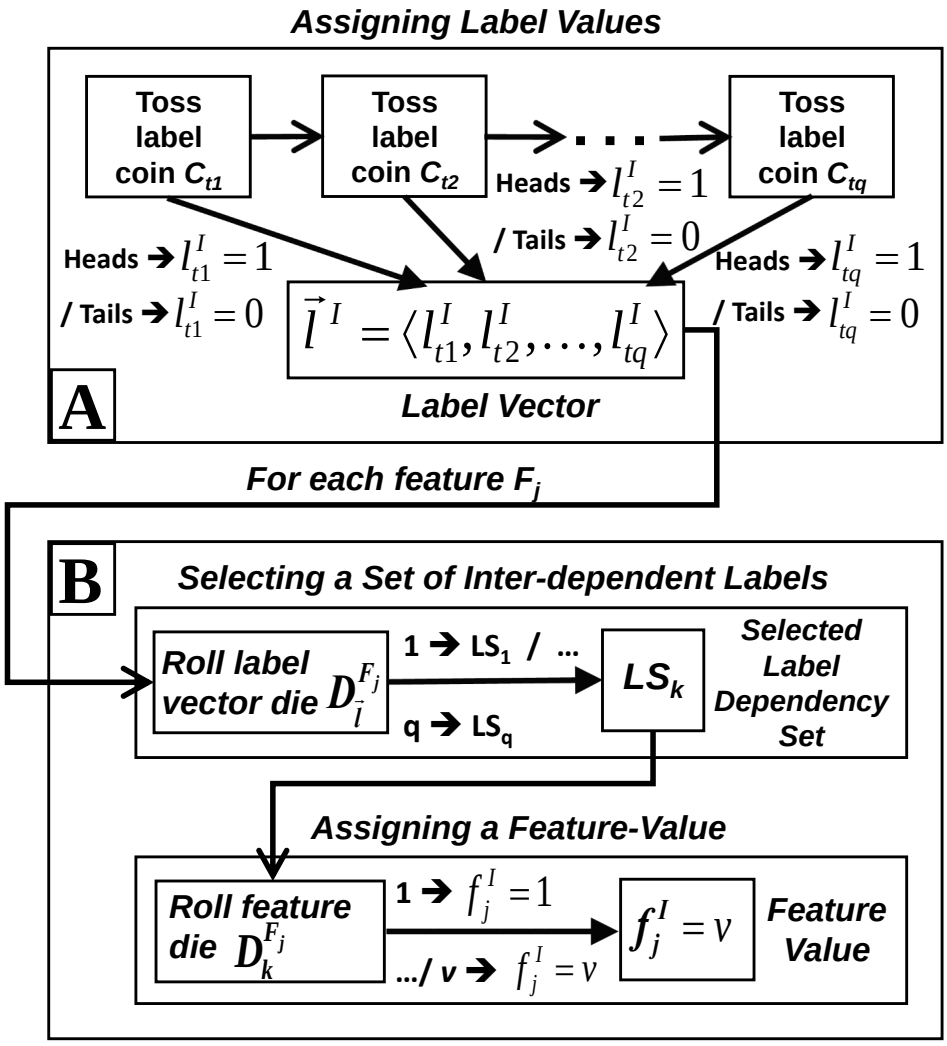


Figure 4.3: The generative process for an instance $\boldsymbol{I}$. First, label coins, $C_{t1}, \ldots, C_{tq}$, are tossed (box A, top); based on the outcomes, label values, $l^I_{t1}, \ldots, l^I_{tq}$, are assigned. Collectively, these values make up the label-vector $\vec{l}^I$ (box A, bottom). For each feature F_j, the label-vector die $D^{F_j}_{\vec{l}}$ is then tossed to select a label dependency set (box B, top); based on the label values in the selected set LS_k, the feature die $D^{F_j}_k$ is tossed to select a feature-value f^I_j (box B, bottom).

of each other given the instance's label vector $\vec{l^I}$:

$$\Pr(\vec{f^I}|\vec{l^I}) = \prod_{j=1}^{d} \Pr(f_j^I|\vec{l^I}) \ . \tag{4.3}$$

Although this assumption over-simplifies feature inter-dependencies, it has been proven effective [1, 119].

(ii) Given the values taken by a label variable L_k and its parents $Pa(L_k)$ in a *selected* label dependency set LS_k for an instance I, a value f_j^I of a feature is *conditionally independent* of all other labels of I:

$$\begin{aligned} \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_1 = l_1^I, \ldots, L_q = l_q^I) = \\ = \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{\mathrm{Pa}(L_k)}) \ . \end{aligned} \tag{4.4}$$

Figure 4.4 shows a graphical representation of the generative mixture model presented above. Nodes represent random variables, and directed edges represent dependencies among variables. Label and feature random variables are denoted as circles. In contrast, the label dependency set, LS_k is denoted by a square as its value is deterministically assigned based on values taken by the label variable L_k and its parents $\mathrm{Pa}(L_k)$. Notably, the node for LS_k represents a set, and as such the directed edge from LS_k to the feature F_j is a short-hand for multiple edges connecting each label variable in LS_k with F_j as described in Section 4.2.1. Variables representing labels and features are *observed*, i.e., their values are provided within the training dataset. These variables are shown as shaded in the figure. The value of the variable λ^{F_j} is governed by the mixture parameter $\theta_j^{\vec{l}}$ and is not given as part of the dataset. As such, it is *latent* and shown as unshaded.

Under all the above mentioned independence assumptions and based on the structure of our generative model, the joint probability of the label vector $\vec{l^I}$ and the feature vector $\vec{f^I}$ is expressed as:

$$\begin{aligned} \Pr(\vec{l^I}, \vec{f^I}) &= \Pr(\vec{l^I}) \Pr(\vec{f^I}|\vec{l^I}) = \quad \text{(Chain rule)} \\ &= \prod_{i=1}^{q} \Pr(l_i^I|\mathcal{V}_{\mathrm{Pa}(L_i)}) \Pr(\vec{f^I}|\vec{l^I}) = \quad \text{(Eq. 4.1)} \end{aligned}$$

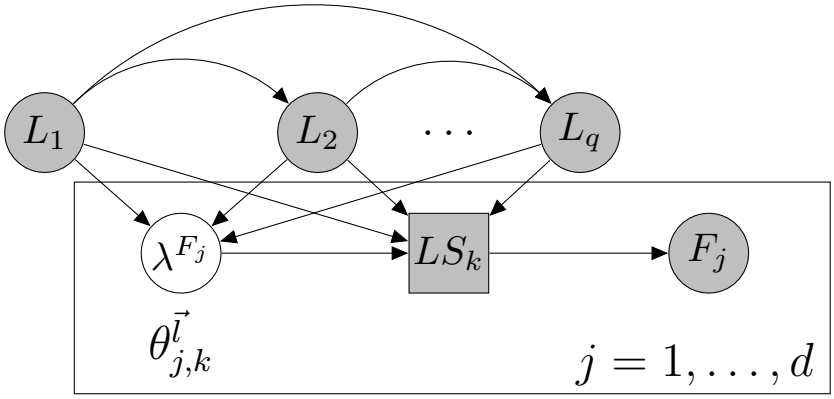


Figure 4.4: Bayesian network representing the generative mixture model of multi-label data. Nodes represent random variables. Directed edges represent dependencies between variables. Label and feature variables are shown as circles, and the label dependency set (LDS) is shown as a square. Shaded nodes represent observed variables and unshaded nodes represent latent variables. The variable λ^{F_j} takes on a value k with a probability $\theta^{\vec{l}}_{j,k}$, which indicates the selection of an LDS LS_k. The rectangular plate notation is used to represent replication of features and LDS with the same dependencies.

$$
\begin{aligned}
&= \prod_{i=1}^{q} \Pr(l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)}) \times \prod_{j=1}^{d} \Pr(f_j^I | \vec{l^I}) = \quad \text{(Eq. 4.3)} \\
&= \prod_{i=1}^{q} \Pr(l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)}) \times \prod_{j=1}^{d} \sum_{k=1}^{q} \Pr(f_j^I, \lambda^{F_j}{=}k | \vec{l^I}) = \quad \text{(marginalizing over all possible label dependency sets)} \\
&= \prod_{i=1}^{q} \Pr(l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)}) \times \prod_{j=1}^{d} \sum_{k=1}^{q} \Pr(f_j^I | \lambda^{F_j}{=}k, \vec{l^I}) \Pr(\lambda^{F_j}{=}k | \vec{l^I}) = \quad \text{(Chain rule)} \\
&= \prod_{i=1}^{q} \Pr(L_i{=}l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)}) \\
&\qquad \times \prod_{j=1}^{d} \sum_{k=1}^{q} \theta_{j,k}^{\vec{l}} \Pr(F_j{=}f_j^I | \lambda^{F_j}{=}k, L_k{=}l_k^I, \mathcal{V}_{\mathrm{Pa}(L_k)}) \ , \quad \text{(Eq. 4.4)}
\end{aligned}
\tag{4.5}
$$

where each term corresponds to a parameter of the model as described below.

(a) $\prod_{i=1}^{q} \Pr(L_i{=}l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)})$ is the factorization of the joint probability $\Pr(\vec{l^I}){=}\Pr(L_1{=}l_1^I, \ldots, L_q{=}l_q^I)$, based on the individual q label values, given the conditional independencies encoded in the network;

(b) $\Pr(F_j{=}f_j^I | \lambda^{F_j}{=}k, L_k{=}l_k^I, \mathcal{V}_{\mathrm{Pa}(L_k)})$ denotes the conditional probability of a feature value f_j^I ($1 \leq j \leq d$, where d is the total number of features), given the values taken by a label variable L_k and its parents $\mathrm{Pa}(L_k)$; $L_k \cup \mathrm{Pa}(L_k)$ comprises the label dependency set LS_k (under the current generative mixture model);

(c) $\theta_{j,k}^{\vec{l^I}}$ denotes the probability that the label dependency set LS_k is selected given a label-vector $\vec{l^I}$ for a feature F_j.

4.3 Model Learning and Inference

We next introduce a procedure for learning the structure and the parameters of our generative model, and present an inference procedure for multi-label classification.

4.3.1 Structure and Parameter Learning

We employ an iterative procedure to learn the Bayesian network structure, specifically the structure of inter-dependencies among the label nodes shown at the

top of Figure 4.4, and to estimate the model parameters shown on the RHS of Equation 4.5. This iterative procedure is summarized in the pseudocode shown in Figure 4.5.

In each iteration, we first learn a label inter-dependency structure using the BANJO package [96]; we then estimate the model parameters through an Expectation Maximization process; following that, we infer multi-label values for instances in the training set. The inter-dependency structure is learned in the first iteration from the training-set labels, and in subsequent iterations, from the most recently inferred label values. The model parameters are estimated throughout the learning procedure using the training-set labels.

We use this iterative process as it modifies the network structure to reflect inter-dependencies among the most recently inferred label values. We expect such a network to allow the system to capture specific feature-label correlations and conditional independencies, which in turn, may improve the accuracy of the updated label assignments. As shown in the experimental results reported in the next chapter, this assumption is indeed supported by the improved performance of our system.

At the end of each iteration we assess the classification performance of our model over the training set; the iterative procedure is terminated when there is no improvement in performance between two successive iterations. For assessing model performance, we utilize the F_1-score metric when using a database of multi-localized proteins, and the *Hamming-Loss* when using other multi-label datasets; these performance measures are described later in Chapter 5, Section 5.1. The number of iterations needed to learn our model, which we denote by t, may vary across different datasets and also depends on the number of class-labels q; in our experiments, the number of iterations did not exceed 10.

We use maximum likelihood estimation to compute the two sets of *observed* model parameters (shown in Equation 4.5): (a) The conditional probability of a label l_i^I given the values taken by L_i's parents, $\alpha_i = \Pr(L_i = l_i^I | \mathcal{V}_{\mathrm{Pa}(L_i)})$ and (b) The conditional probability of a feature value f_j^I given the values taken by all variables in each label dependency set (LDS), LS_k ($1 \leq k \leq q$), $\phi_{j,k}(f_j^I) = \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{\mathrm{Pa}(L_k)})$.

Initialize Bayesian network structure using the BANJO package [96], based on training-set labels;
Initialize model parameters, α_i and $\phi_{j,k}(v)$ using maximum likelihood estimation, and $\theta^{\vec{l}}_{j,k}$ using EM algorithm, based on training-set labels;
Set initial inferred label values (i.e. each l^I_i, $i = 1, \ldots, q$) for each instance $I \in D$ to 0;
Set t to 0 and P to Hamming accuracy (or F_1-score) of *initial* model over training set;
while *True* **do**
    Update Bayesian network structure using BANJO, based on most recently inferred label values;
    Update model parameters, α_i, $\phi_{j,k}(f^I_j)$, and $\theta^{\vec{l}}_{j,k}$, based on training-set labels;
    while *True* **do**
        Infer values taken by random variables in each label dependency set LS_k (see Figure 4.6 for details);
        if *Hamming accuracy (or F_1-score) of model does not improve* **then**
            break;
        end
    end
    Set P' to Hamming accuracy (or F_1-score) of *updated* model over training set;
    if $P' \leq P$ **then**
        break;
    end
    $P \leftarrow P'$; $t \leftarrow t + 1$;
end

Figure 4.5: Summary of model learning.

To estimate the *latent* parameters, namely, the probability of each label dependency set, LS_k, $\theta^{\vec{l}}_{j,k}$, for a given label vector $\vec{l}$ and a feature F_j, we developed an Expectation Maximization algorithm [23]:

Expectation step: For each instance I, we compute the probability of each LDS LS_k, to be selected for feature F_j, that is, $\lambda^{F_j}=k$, given I's label vector $\vec{l}$ and feature-value f^I_j, as:

$$\Pr(\lambda^{F_j}=k|F_j=f^I_j,\vec{l^I}) = \frac{\theta^{\vec{l^I}}_{j,k}\Pr(F_j=f^I_j|L_k=l^I_k,\mathcal{V}_{\mathrm{Pa}(L_k)})}{\sum_{k=1}^{q}\theta^{\vec{l^I}}_{j,k}\Pr(F_j=f^I_j|L_k=l^I_k,\mathcal{V}_{\mathrm{Pa}(L_k)})} .$$

Maximization step: Using the probabilities computed in the Expectation step, we marginalize over all instances in the training set to re-estimate the mixture parameter, $\theta^{\vec{l}}_{j,k}$, for each feature F_j and label vector $\vec{l}$ as:

$$\theta^{\vec{l}}_{j,k} = \frac{\sum_{v_j}\sum_{\{I|\vec{l^I}=\vec{l},f^I_j=v_j\}}\Pr(\lambda^{F_j}=k|F_j=f^I_j,\vec{l^I})\Pr(F_j=f^I_j|\vec{l^I})}{\sum_{k=1}^{q}\sum_{v_j}\sum_{\{I|\vec{l^I}=\vec{l},f^I_j=v_j\}}\Pr(\lambda^{F_j}=k|F_j=f^I_j,\vec{l^I})\Pr(F_j=f^I_j|\vec{l^I})} ,$$

where v_j takes on all possible values for feature F_j.

We denote by $\vec{l^I}_{LS_k}$ the *restriction* of the label vector $\vec{l^I}$ to only those labels that are in the set LS_k. The conditional probability of a feature F_j to be assigned a value v given the values taken by the label variables in the label dependency set, LS_k, $\Pr(F_j=v|\mathcal{V}_{LS_k})$, is calculated as:

$$\Pr(F_j=v|L_k=l_k,\mathcal{V}_{\mathrm{Pa}(L_k)}) = \Pr(F_j=v|\mathcal{V}_{LS_k}) = \\ = \frac{\sum_{\{I|\vec{l^I}_{LS_k}=\vec{l}_{LS_k},f^I_j=v\}}\Pr(\lambda^{F_j}=k|F_j=f^I_j,\vec{l^I})\Pr(F_j=f^I_j|\vec{l^I})}{\sum_{v_j}\sum_{\{I|\vec{l^I}_{LS_k}=\vec{l}_{LS_k},f^I_j=v_j\}}\Pr(\lambda^{F_j}=k|F_j=f^I_j,\vec{l^I})\Pr(F_j=f^I_j|\vec{l^I})} .$$

Throughout the estimation process, we apply Laplace smoothing [81] by adding fractional pseudocounts to observed counts of events to all the parameters to avoid

overfitting. The process of alternating between the Expectation the Maximization steps is carried out until convergence is reached. To determine convergence, we test that changes to the latent parameter values between iterations are smaller than 0.05.

We next present the inference procedure used by our system to assign multiple labels to instances.

4.3.2 Probabilistic Multi-label Classification

Probabilistic inference in the context of multi-label classification (MLC) amounts to assigning the most probable label vector $\vec{l}^I$ to an instance I based on its feature vector $\vec{f}^I$. Inferring the conditional probability, $\Pr(\vec{l}|\vec{f})$ for each label vector $\vec{l}$ requires 2^q calculations, where q denotes the number of labels. To avoid this exponential number of calculations, some current probabilistic methods for MLC assign a value to each label l_i ($1 \leq i \leq q$) such that the conditional probability $\Pr(l_i|\vec{f}^I)$ is maximized (see e.g. [1]). Others estimate the joint probability of the labels, $\Pr(l_1, \ldots, l_q|\vec{f}^I)$ and eventually infer each label value based on estimates of other labels (see e.g. [22]; [119]). These methods typically infer each label value by utilizing a fixed set of feature-label dependencies captured by their respective models.

In contrast, our system iteratively infers values for sets of multiple labels by capturing in each iteration specific feature-label dependencies based on the most recently inferred label values. We assign values to label variables in each label dependency set (LDS) LS_i (see Section 4.2.1 for the LDS definition), such that the conditional probability $\Pr(\mathcal{V}_{LS_i}|\vec{f}^I)$ is maximized.

To ensure that our method is practically applicable, we set a limit on the maximum number of parents, p, per label variable in the network. In the experiments described here, we restrict the dependency-set size to three (i.e. we set $p=2$) because the mean number of labels per dataset is at most three; the number of inference calculations is thus $2^{p+1}q = 2^3q = 8q$, where q ranges between 6 and 27. To gauge the influence of changes to the values of p on classifier performance, we ran experiments by varying the maximum number of parents in the range 1-3 using *Emotions* and *Scene*

datasets, which have a relatively low number of labels. While increasing the value of p leads to a notable increase in the *Subset accuracy* measure of the classifier, there is no significant improvement in the classifier's *Hamming accuracy* measure (see Chapter 5, Section 5.1 for details about these measures). We anticipate that higher values of p can further improve classifier performance when running experiments on datasets with higher numbers of labels.

As our system considers *multiple* dependency structures between features and labels, we expect that setting a relatively low bound on the dependency-set size considered in each structure, as we do here, will still allow the system to capture the significant dependencies and independencies among features and label subsets, even in larger datasets. Moreover, unconditional direct dependencies are not the only ones our model captures. While each label depends on two parent-labels—thus conditionally independent of other labels, indirect inter-dependencies are still captured throughout the network structure. As demonstrated by the results presented in the next chapter (Section 5.2), our utilization of label subsets of even a small size still significantly improves the performance of our system compared to that of current systems.

Given a feature vector $\vec{f}^I$ of an instance I, our task is to predict its label vector $\vec{l}^I$, which involves assigning a 0/1 value to each of its labels l_i^I $(1 \leq i \leq q)$. According to our probabilistic model, since the value of each label variable L_i depends only on values of its parent nodes $Pa(L_i)$ in a Bayesian network setting, for each L_i, we infer the values of variables in the label dependency set, $LS_i = \{L_i\} \cup Pa(L_i)$. To infer these label values, we follow an iterative process, which is summarized in the pseudocode shown in Figure 4.6. In each iteration, for all possible value assignments, l_i and $\mathcal{V}_{\mathrm{Pa}(L_i)}$ to the label variable L_i and its parents, respectively, we calculate the conditional probability: $\Pr(L_i = l_i, \mathcal{V}_{\mathrm{Pa}(L_i)} | \vec{f}^I, \mathcal{V}^I_{\overline{L}_i})$.[1] The value assignment to L_i and to its parents, $Pa(L_i)$, that maximizes this probability is used as their current estimates. We note that label dependency sets do overlap, that is, the value of the same label

[1] Recall that $\overline{L}_i$ denotes the set of all label variables *other than* L_i and $Pa(L_i)$ and that the values taken by the variables in $\overline{L}_i$ is denoted as $\mathcal{V}_{\overline{L}_i}$.

```
1 foreach label dependency set LS_i = L_i ∪ Pa(L_i) do
2     foreach value assignment to L_i and to Pa(L_i) do
3         Calculate conditional probability: Pr(L_i = l_i, V_Pa(L_i) | f⃗^I, V^I_{L̄_i})[1];
4     end
5     Select value assignment that maximizes the above conditional
      probability;
6     Update inferred values for labels in LS_i if classification performance
      over training set improves;
7 end
```

Figure 4.6: Summary of label inference.

variable L_i may be inferred multiple times, once for each dependency set in which it participates. As such, once the value of L_i is inferred within an iteration, it is only going to be updated during the same iteration if this improves the overall predictive performance of the model. While we currently use the standard inference techniques for Bayesian network models [81], there is much room for optimization by using methods for approximate inference that consider only the likely label combinations and fewer label sets, which we shall pursue in the future.

4.4 Discussion

We presented a probabilistic generative mixture model that captures inter-dependencies among labels as well as dependencies between features and labels. Unlike other approaches for multi-label classification (MLC), our model represents conditional independencies of feature values from labels given subsets of other labels, particularly by introducing the concept of *label dependency sets*.

While employing relatively small label dependency sets, we show in the next chapter through our experiments that the performance of our system can still improve upon that of other current MLC systems. In the future, we plan to develop approximate methods for inference by considering only the likely label combinations and fewer subsets of labels to enable the practical use of larger label dependency sets.

In the next chapter, we report results obtained by employing our generative model to perform multi-label classification using a diverse collection of datasets.

Chapter 5

USING THE GENERATIVE MIXTURE MODEL FOR IMPROVED MULTI-LABEL CLASSIFICATION

Employing the probabilistic generative model introduced in the previous chapter, we present here experiments and results over several multi-label datasets. We first apply the MLC system based on our model to predict locations of proteins by utilizing a dataset of multi-localized proteins. We call the location prediction system as *MDLoc*. To demonstrate the wide applicability of the model in the general context, we then apply the MLC system to a number of other MLC tasks. For the latter experiments, we use a collection of *four* standard multi-label datasets described in Section 5.1.

Section 5.1 provides details about the multi-label datasets we use and about the performance evaluation measures. In Section 5.2, we discuss the experimental results. Finally, Section 5.3 concludes and outlines future directions.

5.1 Datasets and Performance Measures

We introduce here two sets of experiments whose results are presented in Section 5.2. In the first set, we employ a protein dataset previously used by Briesemeister et al. [10] to compare the performance of multi-location prediction systems. In the second set, we utilize a collection of multi-label datasets previously used by Alessandro et al. [1] to assess the performance of multi-label classification (MLC) systems. Details of these experiments are provided next.

For the first set of experiments, we use the same protein dataset that we already presented earlier in Chapter 3. The dataset contains *single-localized* proteins, originally published by Höglund et al. [45], and a collection of *multi-localized* proteins, originally published as part of the DBMLoc dataset [121]. Each protein is represented

by *30 features*, and the 9 possible subcellular locations correspond to *9 class-labels*. We compare the performance of our system to that of state-of-the-art multi-location prediction systems as reported by Briesemeister et al. [10], in their assessment of the YLoc$^+$ system [10], including Euk-mPLoc [14], WoLF PSORT [47], and KnowPred$_{\text{site}}$ [57]. According to the methods used in the previous assessment [10], we employ minimal entropy partitioning technique [29] for feature discretization, and stratified 5-fold cross-validation for training/testing the classifiers.

In the second set of experiments, we use the following multi-label datasets: *Emotions* (72 *features*, 6 *labels*), *Scene* (294 *features*, 6 *labels*), *Yeast* (103 *features*, 14 *labels*), and *Genbase* (1186 *features*, 27 *labels*). We assess the performance of our system by comparing it with that of current MLC systems reported by Alessandro et al. [1] in their study including: ensemble of Bayesian networks, namely, *EBN-J / EBN-M* [1], ensemble of chain classifiers [77] using Naïve Bayes (denoted *ECC-NB*) and using J48 (denoted *ECC-J48*), and Binary Relevance using Naïve Bayes (denoted *BR-NB*) [105]. As per the methodology used in the previous study of MLC systems [1], we discretize each real-valued feature into four bins, select features using a correlation-based feature selection technique [111], and employ the stratified 10-fold cross-validation for evaluating system performance. Table 5.1 summarizes the main distinguishing properties of the compared systems.

Under our current unoptimized implementation, wall clock time for model learning using training instances and inferring multi-labels of test instances combined is on the order of several minutes for datasets with a few labels, (lowest being $\leq$10 minutes for *Emotions*), and on the order of hours for datasets with more labels, (highest being $\sim$*20* hours for *Yeast*). We note that while the run-time of the prototypical system grows quadratically with the number of labels, it grows only linearly with the dataset size. For example, the run-time for the *protein multi-location* dataset (containing *8503* instances, with only *9* labels) is about 0.25 of the run-time for the smaller *Yeast* dataset (*2417* instances) that has *13* labels.

Throughout the experiments, we use the exact same evaluation measures applied

Table 5.1: Characteristics of current systems for multi-label classification.

<table>
<tr><th>MLC system</th><th>Captures dependencies among labels</th><th>Captures conditional independence b/w labels and features (given subsets of other labels</th><th>Employs a generative model for data</th></tr>
<tr><td>Our system</td><td rowspan="2">Using a probabilistic graphical model</td><td>Using label dependency sets and a mixture model</td><td rowspan="2">Using Bayesian network</td></tr>
<tr><td>EBN-M/EBN-J</td><td rowspan="3">Do not capture such conditional independence</td></tr>
<tr><td>ECC-NB</td><td rowspan="2">Using classifier chains</td><td>Using näive Bayes</td></tr>
<tr><td>ECC-J48</td><td>No generative model used</td></tr>
<tr><td>BR-NB</td><td colspan="2">No label inter-dependencies represented</td><td>Using näive Bayes</td></tr>
</table>

in the corresponding previous work as described next. For a given instance I, let $M^I = \{c_i \mid l_i^I = 1, \text{ where } 1 \leq i \leq q\}$ be the set of labels associated with I according to the dataset, and let $\hat{M}^I = \{c_i \mid \hat{l}_i^I = 1, \text{ where } 1 \leq i \leq q\}$ be the set of labels assigned to I by a classifier, where each $\hat{l}_i^I$ is a 0/1 label assignment. The *Hamming* (H_{acc}) and the *Subset* (S_{acc}) accuracies used for the evaluation of multi-label prediction systems [1] are computed as:

$$H_{acc} = 1 - \frac{1}{|D|} \times \sum_{I \in D} \frac{1}{|C|} |M^I \Delta \hat{M}^I| \quad \text{and} \quad S_{acc} = \frac{1}{|D|} \times \sum_{I \in D} \mathcal{I}(M^I = \hat{M}^I) \ ,$$

where Δ is the symmetric difference between M^I and $\hat{M}^I$. The *Hamming-Loss* is thus given by: $1 - H_{acc}$. Additionally, the *Multi-label accuracy* (ML_{acc}) and F_1-*label* score used for evaluating multi-location prediction systems [10] are computed as described earlier in Chapter 3:

$$ML_{acc} = \frac{1}{|D|} \times \sum_{I \in D} \frac{|M^I \cap \hat{M}^I|}{|M^I \cup \hat{M}^I|} \quad \text{and} \quad F_1\text{-}label = \frac{1}{|C|} \times \sum_{c_i \in C} \frac{2 \times Pre_{c_i} \times Rec_{c_i}}{Pre_{c_i} + Rec_{c_i}} \ ,$$

where Pre_{c_i} and Rec_{c_i} for label c_i are *adapted* measures of multi-label precision and recall proposed by Briesemeister et al. [10], and are computed as:

$$Pre_{c_i} = \frac{1}{|\{I \in D | c_i \in \hat{M}^I\}|} \times \sum_{I \in D | c_i \in \hat{M}^I} \frac{|M^I \cap \hat{M}^I|}{|\hat{M}^I|} \ , \quad \text{and}$$

$$Rec_{c_i} = \frac{1}{|\{I \in D | c_i \in M^I\}|} \times \sum_{I \in D | c_i \in M^I} \frac{|M^I \cap \hat{M}^I|}{|M^I|} \ .$$

Furthermore, to evaluate the correctness of predictions made for each location, the *standard precision* and *recall* measures for location s_i also presented earlier in Chapter 3, denoted by *Pre-Std*$_{s_i}$ and *Rec-Std*$_{s_i}$, are defined as:

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

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.

Next, we report results obtained by employing our generative mixture model to assign multiple labels to instances using the above datasets.

5.2 Classification Results

We present in this section two sets of results. The first set of results was obtained based on the protein multi-location dataset, and the second set was obtained using the standard multi-label datasets.

5.2.1 Protein Multi-location Prediction

We compare the performance of our system that is based on the generative model we presented in Chapter 4 (denoted MDLoc) with that of existing location prediction systems over the set of multi-localized proteins derived from DBMLoc [121]. We also report experiments using the *combined set* of single and multi-localized proteins. Our analysis includes an examination of the per-location break-up of the results. Additionally, we focus on several specific examples demonstrating the benefit of incorporating location inter-dependencies into our prediction system.

Table 5.2 shows the F_1-*label* score and the *accuracy* obtained by MDLoc compared to those obtained by current multi-location predictors (YLoc$^+$, Euk-mPLoc, WoLF PSORT, and KnowPred$_{\text{site}}$ as reported by [10] in *Table 3*) and by our preliminary system based on a collection of Bayesian network classifiers (denoted BNCs, presented in Chapter 3), using the same set of multi-localized proteins and evaluation measures. The table shows that MDLoc performs better than the current systems, which includes a top-performing system, YLoc$^+$, which captures dependencies specific to location combinations in the training set. In contrast, MDLoc represents inter-dependencies among locations by means of a Bayesian network as well as captures dependencies between features and various combinations of inter-dependent locations.

To illustrate the use of location inter-dependencies, consider the protein *Securin* which is included in our dataset and localizes to both the cytoplasm (*cyt*) and the nucleus (*nuc*). *Securin*, initially present in the cytoplasm, translocates to the

Table 5.2: Prediction results for *multi-localized* proteins only, averaged over 25 runs of 5-fold cross-validation. The table shows overall F_1-*label* scores and overall accuracy (ML_{acc}) obtained using MDLoc, YLoc$^+$[10], Euk-mPLoc [14], WoLF PSORT [47], and KnowPred$_{\text{site}}$ [57]. All values except ours are taken directly from *Table 3* in the paper by Briesemeister et al. [10]; standard deviations are not available there. The highest values are shown in boldface.

Location Prediction System	$\boldsymbol{F_1}$-***label***	$\boldsymbol{ML_{acc}}$
MDLoc	**0.71** ($\pm$ 0.02)	**0.68** ($\pm$ 0.01)
YLoc$^+$	0.68	0.64
Euk-mPLoc	0.44	0.41
WoLF PSORT	0.53	0.43
KnowPred$_{\text{site}}$	0.66	0.63

Table 5.3: Per location prediction results for the same set of *multi-localized* proteins shown in Table 5.2, averaged over 25 runs of 5-fold cross-validation. The table shows *Multilabel-Precision* (Pre_{s_i}) and *Recall* (Rec_{s_i}), as well as *standard precision* ($Pre\text{-}Std_{s_i}$) and *recall* ($Rec\text{-}Std_{s_i}$), for each location s_i. The various per-location scores are obtained using our system, MDLoc, and a top location prediction system, YLoc$^+$ [10]. Results for YLoc$^+$ were reproduced using our 5-way splits. The p-values indicate statistical significance of the differences between the MDLoc scores and the YLoc$^+$ scores. The highest values are shown in boldface. Standard deviations are shown in parentheses.

Location	**MDLoc**	**YLoc$^+$**	***p*-value**	**MDLoc**	**YLoc$^+$**	***p*-value**
	$\boldsymbol{Rec_{s_i}}$			$\boldsymbol{Pre_{s_i}}$		
cyt (2374)	**.750** (± .012)	.712 (± .009)	≪.001	**.911** (± .008)	.893 (± .010)	≪.001
nuc (2115)	**.776** (± .014)	.728 (± .011)	≪.001	**.929** (± .008)	.924 (± .008)	.03
mem (586)	.527 (± .022)	**.543** (± .018)	.01	**.807** (± .036)	.764 (± .029)	≪.001
ex (562)	.547 (± .035)	**.573** (± .026)	.01	**.833** (± .044)	.740 (± .053)	≪.001
mi (360)	.519 (± .026)	**.536** (± .031)	.04	**.832** (± .042)	.765 (± .033)	≪.001
	$\boldsymbol{Rec\text{-}Std_{s_i}}$			$\boldsymbol{Pre\text{-}Std_{s_i}}$		
cyt (2374)	**.817** (± .021)	.786 (± .020)	≪.001	**.942** (± .009)	.935 (± .009)	.01
nuc (2115)	**.746** (± .028)	.684 (± .015)	≪.001	.904 (± .014)	**.914** (± .014)	.02
mem (586)	.588 (± .042)	**.614** (± .042)	.04	**.794** (± .039)	.730 (± .047)	≪.001
ex (562)	.385 (± .058)	**.401** (± .037)	.3	**.830** (± .046)	.771 (± .055)	≪.001
mi (360)	.388 (± .062)	**.429** (± .060)	.03	**.784** (± .057)	.670 (± .055)	≪.001

nucleus and plays in response to DNA damage [7, 53]. While MDLoc assigns the protein to both the *cytoplasm* and the *nucleus*, YLoc$^+$ assigns it to the *nucleus* only. Our system utilizes the dependence between *nucleus* and *cytoplasm* (represented as a directed edge between the two locations) along with various dependencies between protein features and locations to make an accurate multi-location prediction. Dependence between a pair of locations reflects correlation of one location with another, and in this case, it is well known that proteins shuttle between the nucleus and the cytoplasm to aid DNA repair [34]. MDLoc's benefit from capturing the dependence between *cytoplasm* and *nucleus* is also reflected in its significantly higher *Multilabel-Precision* and *Multilabel-Recall* (Pre_{s_i} and Rec_{s_i}, respectively) for the *cyt* and *nuc* proteins as shown in Table 5.3.

As another example, consider *Transforming Growth Factor alpha (TGF-alpha)*, a protein that initiates cell growth [31] and is known to localize to both the extracellular space (*ex*) and the plasma membrane (*mem*). MDLoc correctly assigns both locations to the protein. Here MDLoc employs the well-known dependence between *extracellular space* and *plasma membrane*, as protein movement between subcellular compartments and the extracellular space occur via the plasma membrane [101]. Again, the value of utilizing location inter-dependencies is demonstrated in MDLoc's significantly improved precision in terms of *Multilabel-Precision* (Pre_{s_i}) on the *ex* and *mem* proteins (while still retaining a similar level of recall, Rec_{s_i}, to that of YLoc$^+$).

As an example for MDLoc's ability to handle proteins whose *location combinations* is not included in the training set, consider *Transmembrane emp24 domain-containing protein 7 (emp24)*. It localizes to the endoplasmic reticulum and transports secretory proteins to the golgi complex [6].[1] MDLoc assigns *emp24* to both the *endoplasmic reticulum* and the *golgi complex*, whereas YLoc$^+$ assigns it to the *endoplasmic reticulum* only. MDLoc makes use of the dependency that captures correlation between the *endoplasmic reticulum* and the *golgi complex*, both of which play a role in

[1] The tables shown do not include *endoplasmic reticulum* and *golgi complex* proteins, as the number of proteins from either of these locations in the dataset is very small.

producing intracellular responses to extracellular signals [58]. We thus see that MDLoc is not restricted to predicting only location combinations in the training set.

We next compare results obtained per location by our system with those obtained by a top-performing current system as well as by our initial system. We show results for locations that have relatively a large number of associated multi-localized proteins. Our comparison reports for each location s_i, the *Multilabel-Precision* (Pre_{s_i}) and *Multilabel-Recall* (Rec_{s_i}) as well as the *standard precision* ($Pre\text{-}Std_{s_i}$) and *recall* ($Rec\text{-}Std_{s_i}$).

Table 5.3 shows per-location prediction results obtained using MDLoc compared to those obtained using YLoc$^+$[10] for multi-localized proteins. Prediction results for the other current systems are not shown here as they are not publicly available. For the cytoplasm and the nucleus, which have a large number of proteins, the precision and recall values obtained using MDLoc are significantly higher in most cases than those obtained using YLoc$^+$. For locations with much fewer proteins, while the recall values when using MDLoc are marginally lower than when using YLoc$^+$, MDLoc's precision values are typically significantly higher than those of YLoc$^+$. We note that YLoc$^+$ assigns each protein to *all* the locations whose probability exceeds a pre-defined threshold; as such, the number of locations it assigns exceeds that to which the protein actually localizes resulting in a lower precision. In contrast, MDLoc does not simply assign a protein to the most probable location, but rather, it simultaneously considers a *set* of locations and assigns each protein to the set whose overall probability is the highest, leading to a higher precision.

Table 5.4 shows the per-location prediction results obtained by MDLoc, in comparison to those obtained by BNCs on the *combined dataset of both single- and multi-localized proteins*. While MDLoc's precision values are somewhat lower than those of BNCs, MDLoc's recall is typically higher. MDLoc simultaneously infers the probability of a *set* of locations; in contrast, BNCs use an independent Bayesian network structure to infer the probability of each location separately. As such, the likelihood of BNCs to correctly assign the combination of several locations to a protein is much lower than

Table 5.4: Per location prediction results for the *combined set* of single- and multi-localized proteins, averaged over 25 runs of 5-fold cross-validation. The table shows the same measures shown in Table 5.3 obtained using our two systems, MDLoc and BNCs. The p-values indicate statistical significance of the differences between the MDLoc scores and the BNCs scores. The highest values are shown in boldface. Standard deviations are shown in parentheses.

Location	**MDLoc**	**BNCs**	***p*-value**	**MDLoc**	**BNCs**	***p*-value**
	$\boldsymbol{Rec_{s_i}}$			$\boldsymbol{Pre_{s_i}}$		
cyt (2374)	**.825** (± .009)	.795 (± .011)	≪.001	**.819** (± .013)	.809 (± .018)	.03
nuc (2115)	**.830** (± .010)	.784 (± .017)	≪.001	.822 (± .014)	**.832** (± .013)	.02
mem (586)	**.780** (± .020)	.737 (± .022)	≪.001	.864 (± .020)	**.912** (± .019)	≪.001
ex (562)	**.822** (± .012)	.780 (± .014)	≪.001	.872 (± .014)	**.900** (± .012)	≪.001
mi (360)	**.773** (± .013)	.730 (± .025)	≪.001	.861 (± .024)	**.885** (± .023)	.001
	$\boldsymbol{Rec\text{-}Std_{s_i}}$			$\boldsymbol{Pre\text{-}Std_{s_i}}$		
cyt (2374)	**.867** (± .015)	.861 (± .014)	.1	**.854** (± .014)	.840 (± .011)	.001
nuc (2115)	**.808** (± .021)	.736 (± .031)	≪.001	.783 (± .020)	**.786** (± .026)	.6
mem (586)	**.715** (± .030)	.652 (± .024)	≪.001	.839 (± .028)	**.906** (± .022)	≪.001
ex (562)	**.842** (± .017)	.805 (± .017)	≪.001	.882 (± .014)	**.900** (± .015)	≪.001
mi (360)	**.719** (± .028)	.664 (± .034)	≪.001	.843 (± .026)	**.873** (± .034)	.001

its probability to correctly assign a single location, which directly translates into a relatively low recall measure. When using MDLoc, the increase in recall values for almost all cases is higher than the decrease in the precision values, except in the case of the *extracellular space (ex)*. Notably, proteins in the extracellular space all originate from or are bound toward another location within the cell and as such predicting them as extracellular is challenging for most prediction systems.

Moreover, MDLoc assigns some proteins hitherto known to localize only to a single location into multiple locations. It is likely that at least some of these additional predicted locations are indeed correct and can be the subject of an experimental validation. For instance, *Calreticulin (Cal)* is currently annotated by SwissProt as localized to the endoplasmic reticulum only. However, MDLoc assigns the protein to both the *endoplasmic reticulum* and the *extracellular space*, and work by Gold et al. [37] suggests that *Cal* indeed relocates from the *endoplasmic reticulum* to the *extracellular space*.

To demonstrate improved prediction performance of our system across locations regardless of the number of associated proteins, we examine the statistically significant differences in the *Multilabel-Recall* for the location with the highest number of multi-localized proteins (*cytoplasm*, 2,374 proteins) and for the location with the lowest number (*endoplasmic reticulum*, 115 proteins). The *Multilabel-Recall* for *cytoplasm* (Rec_{cyt}) increases from 0.80 when classifying using BNCs, to 0.83 when classifying using MDLoc. Similarly, the *Multilabel-Recall* for *endoplasmic reticulum* (Rec_{ER}, not shown in Table 5.4) increases from 0.64 to 0.69. This analysis demonstrates the advantage of using MDLoc for predicting protein locations, not just for locations that have a large number of associated proteins but also for locations that are associated with relatively few proteins.

Table 5.5 shows the prediction results per location combination obtained using MDLoc in contrast to those obtained using BNCs for all location combinations, using multi-localized proteins only. For each location-combination in the dataset, we show the number of proteins with correct predictions for both locations, as well as for the first of the two locations, and for the second, separately. For almost all combinations,

Table 5.5: Per location-combination prediction results for *multi-localized* proteins only, obtained using one run of 5-fold cross-validation. For each combination of *two* locations, the table shows the number of proteins with correct predictions for both locations, for the first of the two locations, and for the second of the two locations, using MDLoc and using BNCs. The highest values are shown in boldface.

Location Combination	**MDLoc**	**BNCs**	**MDLoc**	**BNCs**	**MDLoc**	**BNCs**
	Both locations correct		**1st location correct**		**2nd location correct**	
cyt_nuc (1882)	**1253 (66.6%)**	976 (51.9%)	**1603 (85.2%)**	1578 (83.8%)	**1481 (78.7%)**	1240 (65.9%)
ex_mem (334)	**34 (10.2%)**	16 (4.8%)	**87 (26%)**	60 (18%)	**258 (77.2%)**	246 (73.7%)
cyt_mem (252)	**31 (12.3%)**	15 (6%)	**186 (73.8%)**	174 (69%)	**82 (32.5%)**	68 (27%)
cyt_mi (240)	**36 (15%)**	25 (10.4%)	164 (68.3%)	**165 (68.8%)**	**99 (41.3%)**	85 (35.4%)
nuc_mi (120)	**15 (12.5%)**	11 (9.2%)	**43 (35.8%)**	37 (30.8%)	**67 (55.8%)**	64 (53.3%)
ER_ex (115)	**35 (30.4%)**	16 (13.9%)	**66 (57.4%)**	**66 (57.4%)**	**51 (44.3%)**	27 (23.5%)
ex_nuc (113)	51 (45.1%)	**54 (47.8%)**	**73 (64.6%)**	68 (60.2%)	**72 (63.7%)**	68 (60.2%)

the number of proteins whose location is correctly predicted by MDLoc is significantly higher than the corresponding number when using BNCs. For instance, the number of *multi-localized* proteins correctly assigned to the location-combination is *cytoplasm* and *nucleus* increases significantly from *976* when using BNCs, to *1253* when using MDLoc. The increase shows that location inter-dependencies learned using MDLoc help to improve predictions for multi-localized proteins.

5.2.2 The General Case of Multi-label Classification

The second set of experiments are performed over the standard multi-label datasets. We compare the performance of our system based on a generative model with that of current multi-label classification (MLC) systems.

Table 5.6 shows the *Hamming* and the *Subset* accuracies (H_{acc} and S_{acc}, respectively) of our system compared to that obtained by current MLC systems (as reported by Alessandro et al. [1], *Tables 2, 3, 4, 6* there), obtained over the same multi-label datasets and using the same evaluation measures. The results show that our system has higher H_{acc} and S_{acc} than all other systems over all datasets except *Genbase*. The differences in the improved performance values are statistically significant ($p \ll 0.05$, according to the *2-sample t-test* [21]). Over the *Genbase* dataset, our system has the same H_{acc} as the others and a slightly lower S_{acc}, although the latter difference is not statistically significant. This decrease in the performance of our system can be attributed to the fact that instances in the *Genbase* dataset are typically associated with much fewer labels than those in the other datasets. As a result, our system captures fewer dependencies and independencies among labels and features in this dataset.

5.3 Summary

We presented multi-label classification (MLC) experiments over a diverse collection of datasets by employing our generative model presented in Chapter 4. The experiments show that utilizing the intricate dependence and independence structure

Table 5.6: Prediction results over standard multi-label datasets, obtained using one run of 10-fold cross-validation. The table shows Hamming and Subset accuracies, H_{acc} and S_{acc}, obtained using multi-label classification (MLC) systems, namely, our system, EBN-M/EBN-J [1], ECC-J48 [77], ECC-NB [77], and BR-NB [105]. All values except ours are taken directly from *Tables 2, 3, 4, 6* in Alessandro et al. [1]. Highest values are shown in boldface. Standard deviations are shown in parenthesis.

MLC System	**Emotions**	**Scene**	**Yeast**	**Genbase**
	$\boldsymbol{H_{acc}}$			
Our system	**.793** (± .021)	**.898** (± .010)	**.786** (± .007)	**.998** (± .001)
EBN-M / EBN-J	.780 (± .022)	.880 (± .010)	.773 (± .008)	**.998** (± .001)
ECC-J48	.780 (± .027)	.883 (± .008)	.771 (± .007)	**.998** (± .001)
ECC-NB	.781 (± .026)	.835 (± .007)	.703 (± .009)	.996 (± .001)
BR-NB	.776 (± .023)	.826 (± .008)	.703 (± .011)	.996 (± .001)
	$\boldsymbol{S_{acc}}$			
Our system	**.319** (± .036)	**.610** (± .030)	**.158** (± .029)	.956 (± .022)
EBN-M / EBN-J	.263 (± .062)	.575 (± .030)	.127 (± .018)	**.965** (± .015)
ECC-J48	.260 (± .038)	.531 (± .038)	.132 (± .023)	.934 (± .015)
ECC-NB	.295 (± .060)	.294 (± .022)	.102 (± .023)	.897 (± .031)
BR-NB	.261 (± .049)	.276 (± .017)	.091 (± .020)	.897 (± .0031)

among features and labels captured by our model contributes to improved accuracy in multi-label classification, compared to a variety of state-of-the-art systems.

Our system incorporates location inter-dependencies into the process of assigning locations for proteins. We showed examples of such dependencies that our model utilizes to achieve improved performance over current top systems such as YLoc+ [10]. Moreover, our system improves upon the performance of current MLC systems while performing a number of MLC tasks, such as predicting emotions of songs, predicting labels of scenes, thus demonstrating the model's wide applicability in the general context of multi-label classification.

Unlike current MLC systems, our model captures dependencies between feature values and subsets of labels and conditional independencies of feature values from certain labels given subsets of other labels. As an example, in the *Emotions* dataset [103], the *tone* feature of songs depends on the class labels *Quiet-Still*, *Sad-Lonely*, *Amazed-Surprised*, and *Angry-Aggressive*. Typically, songs labeled as belonging to the first two classes have a *Low* tone while those in the last two classes have a *High* tone. Our system directly captures the conditional independence of the tone feature from the first two labels given the other two labels. In contrast, current systems do not attempt to capture such subtle and informative dependencies and independencies.

In the next chapter, we present a procedure to construct an extensive collection of multi-localized proteins. The set of multi-localized proteins we used earlier in our experiments is limited to proteins localizing to only *two* locations and does not contain information from many up-to-date repositories.

Chapter 6

AN EXTENSIVE COLLECTION OF MULTI-LOCALIZED PROTEINS

The set of multi-localized proteins that we used in the experiments presented earlier in Chapters 3 and 5 is limited to proteins that localize to only *two* locations. This set is derived from the DBMLoc dataset and is the most comprehensive collection of multi-localized proteins currently available. However, protein information stored in the DBMLoc-derived set is not up-to-date, and it does not contain details about underlying cell characteristics such as tissue, cell line, and disease.

In this chapter, we present a procedure we have devised to construct a database that contains eukaryotic proteins from complex multicellular organisms, in addition to those from other simpler organisms. The procedure allows us to extract reliable information about proteins localizing to *multiple* locations, including cell characteristics from a number of online repositories that store up-to-date experimental evidence for the locations.

In Section 6.1, we provide further details about existing repositories that store information about multi-localized proteins. Section 6.2 briefly reviews sources of protein information that are currently available. Section 6.3 describes the methodology used to construct a new database of multi-localized proteins, and provides a summary of the contents in this database.

6.1 Motivation

Subcellular locations of proteins are determined using experimental methods, and may also be predicted using similarity search and machine learning methods. Locations obtained through experimental methods are typically highly reliable. While

several databases contain information about proteins, very few of them store entries of eukaryotic proteins that localize to multiple compartments.

Organelle DB [112, 113] stores information about single and multiple locations of eukaryotic proteins. However, some protein locations are not experimentally verified. Furthermore, while protein-entries stored in this repository are related to model organisms such as yeast, roundworm, fruit fly, mouse, and arabidopsis, information about human proteins is not available.

The most comprehensive current set of multi-localized proteins, which was derived from the DBMLoc dataset [121], contains information about prokaryotic and eukaryotic proteins that localize to only *two* locations. This set contains DBMLoc proteins whose sequences share less than 80% sequence similarity with each other, and was built by Briesemeister el al. [10] to assess the performance of several multi-location prediction systems. We note that the initial release of the complete DBMLoc dataset (containing 10,470 proteins as reported by Zhang et al. [121]) is no longer publicly available at the time of writing this thesis. While DBMLoc contains protein-records from a number of databases, including Organelle DB, the dataset does not contain information from recently published repositories and from data stores that are constantly updated. Moreover, many proteins locations stored in DBMLoc are not experimentally determined but are based on locations of interacting proteins.

Notably, functions of proteins may depend on underlying cell characteristics including tissue, cell line, and disease. However, none of the above mentioned databases stores information about these cell characteristics.

We thus present a procedure to build a database that stores reliable information about eukaryotic proteins found in complex multicellular organisms in addition to those found in much simpler organisms. Relevant characteristics of the cell are also stored. Our procedure extracts entries corresponding to *multi-localized* proteins from organisms including human, mouse, and Arabidopsis, using *five* up-to-date repositories.

We next discuss these repositories and explain the reasons behind using them to construct our database.

6.2 Protein Data Sources

Several current databases store information about locations of proteins. Some of these store protein locations that are determined using experimental methods such as mass spectrometry [25] and green fluorescence detection [42]; namely, the Human Protein Reference Database [74], the Human Protein Atlas (HPA) [107], LOCATE [30, 97], and SUBA3 [100]. Others store protein locations inferred using computational methods such as similarity search (e.g. DBSubLoc [39]) and machine learning based approaches (e.g. eSLDB [4]). UniprotKB [2] is the most comprehensive publicly available protein database that stores locations obtained using experimental methods as well as using computational approaches. While experimental methods are most reliable for identifying protein locations, inferring locations of proteins via computational methods is not considered to be as reliable.

As our goal is to construct a database that stores reliable information about multi-localized proteins, we use in this work only resources that contain experimentally verified locations of proteins. Since locations of proteins depend on underlying characteristics of the cell, we also utilize resources that contain details about cell characteristics such as tissue, cell line, and cell type, as well as references to the literature that validates the information we store. A preliminary collection of protein resources that are publicly available was established in an earlier study [67]. We select from there the following repositories for our study: SUBA3 [100], the Human Protein Reference Database (HPRD) [74], the Human Protein Atlas (HPA) [107], LOCATE [30, 97], and UniprotKB [2]. We focus on these five sources to construct our database of multi-localized proteins, as these sources contain experimentally verified information about eukaryotic proteins that is up-to-date.

SUBA3 stores experimentally determined locations for Arabidopsis proteins. On the other hand, both HPRD and HPA store subcellular locations exclusively for human proteins. For most of the human proteins stored in HPRD, the locations have been obtained through manual curation of experimental data in the literature. Additionally, HPRD integrates experimentally verified location annotations obtained from Human

Proteinpedia [52], a distributed protein annotation system. HPA stores locations for proteins in normal and cancer tissues obtained through immunofluorescence, which is a microscopy-based experimental method.

LOCATE stores subcellular locations of mouse and human proteins. The location annotations in LOCATE have been obtained through manual curation of the literature as well as through experimental methods. Finally, UniprotKB contains information of proteins from several organisms. We utilize a particular section of UniprotKB, namely, Swiss-Prot [9], that typically stores locations of proteins that are experimentally validated and/or manually curated.

6.3 Database Construction

The construction procedure for our protein database constitutes two main steps: 1) Extraction of protein information from the online repositories discussed above; 2) Development of the database to store protein information. We describe the details of each step in the following sections.

6.3.1 Protein Extraction

We downloaded each of the five public databases mentioned above, and extracted experimentally verified locations of proteins as well as details about underlying cell characteristics such as tissue, cell line, and disease. Table 6.1 summarizes the information extracted from all the data sources. The methodology used for extracting relevant data is explained below.

SUBA3 [100]: This online data repository was constructed using The Arabidopsis Information Resource (TAIR) [99], and provides information about *Arabidopsis* proteins. We downloaded 10,253 Arabidopsis proteins and their experimentally determined locations from SUBA3 as a delimited text file. The file consists of protein attributes including identifiers, descriptions, molecular weights, expression levels, locations that are experimentally verified or that are computationally predicted, and experimental data that have been manually curated from literature sources.

Table 6.1: Summary of the information extracted from protein data sources.

Data Sources	Details	Number of Proteins	Extracted Attributes
SUBA3	Arabidopsis proteins extracted from the Arabidopsis information resource	10,253	Protein name; Swiss-Prot ID; Arabidopsis gene ID; Gene name; Subcellular locations; Amino-acid sequence; PubMed IDs
HPRD	Human proteins extracted from a number of external sources	21,463	Protein name; Swiss-Prot ID; RefSeq ID; OMIM ID; Entrez gene ID; Gene name; Primary & alternate subcellular locations; Amino-acid sequence; PubMed IDs
HPA	Human proteins from normal and cancer tissues detected using immunohistochemistry	9,963	Protein name; Swiss-Prot ID; Ensembl gene ID; Gene name; Main & additional subcellular locations; Tissues & cell types; Cell lines and locations; Amino-acid sequence
LOCATE	Human and mouse proteins detected using immunofluorescence	Mouse – 20,678 and Human – 21,292	Protein name; Swiss-Prot ID; Entrez protein ID; RefSeq ID; Ensembl gene ID; Entrez gene ID; Gene name; Subcellular locations; Cell lines and locations; Amino-acid sequence; PubMed IDs
Uniprot-KB / Swiss-Prot	Proteins from a number of organisms aggregated from a number of sources	145,763	Protein name; Swiss-Prot ID; Ensembl protein ID; RefSeq ID; Ensembl gene ID; Gene name; Subcellular locations; Amino-acid sequence; PubMed IDs

We extracted the following attributes for each protein: name, Swiss-Prot ID, Arabidopsis gene ID (AGI), gene name according to TAIR, locations and relevant experimental methods, amino-acid sequence, and PubMed IDs of articles that identify experimental methods used to determine protein locations. For proteins whose entries miss information, we used the UniProtKB/Swiss-Prot resource [9] to retrieve gene names based on their Swiss-Prot IDs, and the TAIR [99] to retrieve Swiss-Prot IDs and/or amino-acid sequences based on their AGIs.

The Human Protein Reference Database (HPRD) [70, 74]: This online database stores information about *Human* proteins. HPRD utilizes a number of external resources to aggregate protein information. For example, the Online Mendelian Inheritance in Man (OMIM) repository [40], which is a knowledge base of human genes and genetic disorders, is used to associate proteins with disease annotations; the RefSeq database [75] is used to link proteins from other databases; the Entrez database [59] is used to retrieve gene information of proteins. Furthermore, HPRD integrates a distributed annotation system that allows scientists to submit experimental data. The annotation system is used by researchers to populate HPRD with new proteins.

We downloaded 21,463 human proteins and their experimentally determined locations from HPRD as XML files. In addition to the protein attributes mentioned above, each file contains primary and alternate subcellular locations, functions, disease associations, tissue expressions, and references to published literature that describe experiments identifying protein locations. While *primary* locations hold a protein's most widely known locations, *alternate* holds the protein's lesser known locations. For example, HPRD stores *cytoplasm* as the primary location of *GRB2*, a growth factor protein, and *nucleus* as the protein's alternate location. It is well known that *GRB2* localizes to the *cytoplasm*. However, only a careful search revealed that the protein also localizes to the *nucleus* [69].

We extracted the following attributes for each protein: name, IDs (e.g. Swiss-Prot ID, RefSeq ID), gene IDs (e.g. OMIM ID, Entrez ID), gene name, primary and

alternate locations, amino-acid sequence, and PubMed IDs of articles that identify protein locations. For proteins that miss Swiss-Prot IDs, we utilized the UniProtKB/Swiss-Prot resource [9] to retrieve this information based on their RefSeq IDs.

The Human Protein Atlas (HPA) [107]: This online database also stores information about *Human* proteins. HPA stores expression and localization profiles of human proteins in normal as well as in cancer tissues. Additionally, the database contains high-resolution images obtained from expression studies that employ experimental methods such as immunohistochemistry [110].

We downloaded 9,963 human proteins and their experimentally determined locations from HPA as an XML file. In addition to protein attributes listed earlier, this file contains classes (assigned based on functions, disease associations, evidence), main and other subcellular locations, tissue expressions, cell lines, and tissue images. *Main* locations of a protein store locations with the strongest staining evidence, whereas *other* locations store those supported by much weaker staining.

We extracted the following attributes for each protein: name, Swiss-Prot ID, Ensembl gene ID, gene name, main and other locations, tissues and cell types, cell lines and locations, and amino-acid sequence. For proteins whose entries miss information, we used the HUGO Gene Nomenclature Committee (HGNC) resource [84] to retrieve gene names, and the UniProtKB/Swiss-Prot database [9] to retrieve sequences based on their Swiss-Prot IDs and/or gene names.

LOCATE [30, 97]: This online repository stores information about both *Mouse* and *Human* proteins. LOCATE contains locations of proteins determined using experimental imaging methods such as immunofluorescence. The database also stores locations obtained from manually curated literature reviews. Furthermore, some of the localization data in LOCATE is obtained from external resources such as LIFEdb [5], Mouse Genome Informatics [28], UniProt [2], and ENSEMBL [49].

We downloaded proteins and their locations that are either experimentally determined or are manually curated from LOCATE as two XML files. One file consists of 20,678 mouse proteins, and the other file consists of 21,292 human proteins. Manual

location annotations of proteins were constructed using experimental evidence reported in peer-reviewed publications.

We extracted the following attributes for each protein: name, IDs (e.g. Swiss-Prot ID, Entrez ID, RefSeq ID), gene IDs (e.g. Ensembl ID, Entrez ID), gene name, subcellular locations, cell lines and locations, amino-acid sequence, and PubMed IDs of articles that identify protein locations. For proteins whose entries miss information, we used the HGNC resource [84] to retrieve gene names based on a combination of their Swiss-Prot IDs, RefSeq IDs, Ensembl gene IDs, and Entrez gene IDs. Furthermore, we used the UniProtKB/Swiss-Prot database [9] to retrieve protein and gene IDs.

UniProtKB/Swiss-Prot [9]: This online repository stores protein information from a number of organisms, and references several existing databases. Protein locations in Swiss-Prot are typically validated experimentally or retrieved from manually curated literature sources. We downloaded 145,763 eukaryotic proteins and their highly reliable locations from Swiss-Prot as an XML file, and extracted the following attributes for each protein: name, IDs (e.g. Swiss-Prot ID, Ensembl ID, RefSeq ID), Ensembl gene ID, gene name, subcellular locations, amino-acid sequence, and PubMed IDs of articles that identify protein locations. While proteins may be associated with multiple Swiss-Prot IDs, we select only the primary IDs of proteins.

Protein locations in Swiss-Prot are associated with qualifiers that capture reliability of the annotations. For example, "*Potential*" indicates that the annotation was computationally predicted; "*Probable*" suggests that the annotation may be supported by indirect experimental evidence; "*By Similarity*" indicates that the annotations are based on locations of proteins with similar amino-acid sequences. To ensure that the annotations we extract are reliable, protein locations that are non-experimentally obtained and are associated with the above qualifiers are excluded.

Data Extraction. We implemented Python programs to extract protein information from the *five* repositories described above. We extracted data from different sources that were available in the form of XML files and delimited text files, and aggregated only attributes relevant to our study.

In the next section, we describe the procedure used to construct a collection of multi-localized proteins.

6.3.2 Database Design and Implementation

To develop a database of proteins, we identified entities in the protein data and relationships among these entities, and designed an entity-relationship model for the data. We implemented the model as a MySQL relational database, and inserted protein information aggregated from the sources discussed in Section 6.3.1 into the database. Details about the entities and the relationships represented and about the database schema are presented next.

Entities and relationships

As we are building a database of proteins, a protein is the most basic entity in our study. Other entities included in the database are gene, sequence, location, publication, tissue, and cell line. We store information about these entities per protein. It is known that specific sub-sequences of nucleotides in genes by mechanisms such as alternative splicing code for different proteins. Thus, a gene can be associated with one or more proteins, and a protein can be associated with one or more genes. We describe the relationship between any two entities by specifying the maximum number of times an instance of one entity type can be associated with instances of a related entity type, which is referred to as the relationship *cardinality*. The cardinality of the relationship between genes and proteins, is denoted as *many-to-many* since instances in each of these two entities can be associated with many instances in the other.

A protein consists of a sequence of amino acids. Furthermore, different forms of the same protein comprise sequences that are similar to each other and are known as isoforms of the protein. As each protein can be associated with one or more sequences, proteins and sequences share a *one-to-many* relationship. This relationship is mandatory since a protein is always associated with at least one sequence. Similarly, proteins and locations also share a mandatory *one-to-many* relationship, since we consider in

this study only proteins whose locations are available. In contrast, proteins share optional *one-to-many* relationships with other entities including tissues, cell lines, and publications since we store information about these entities only when available.

Figure 6.1 shows the entity relationship diagram for our data model. Entities are represented as rectangles, and relationships between entities are shown as dotted lines. The relationship *cardinality* is shown by combinations of a single slash and a crow's-feet. For instance, a single slash closest to *Protein* and a crows-feet closest to *Sequence* together indicate a *one-to-many* relationship between the two entities; two crow's-feet closest to the entities, *Gene* and *Protein* indicate a *many-to-many* relationship. An optional relationship is shown using an oval. For instance, the relationship between entities, *Location* and *Tissue* is optional as tissue information is not available for all proteins in the dataset.

Relational schema

Utilizing the entity relationship model described above, we created one table per entity (primary keys are given in parenthesis): *Protein* (*database_id*), *Gene* (*gene_id*), *Sequence* (*seq_id*), *Location* (*location_id*), *CellLine* (*cellline_id*), *Tissue* (*tissue_id*), and *Publication* (*publication_id*).

Each *one-to-many* relationship between a pair of entities is captured using a *foreign key* in the table corresponding to one entity that references the *primary key* in the table corresponding to a related entity. For example, *seq_id* is a foreign key in the tables, *Location* and *Publication*, and references in each case the primary key in the table, *Sequence*. As another example, *location_id* is a foreign key in *CellLine*, *Tissue*, and *Publication*, and references in each case the primary key in *Location*.

We created an additional table, *CodedBy* (*database_id* and *gene_id*), to capture the *many-to-many* relationships between the entities, *Protein* and *Gene*. This table provides a mapping between primary keys in the *Gene* and *Protein* tables. Detailed table definitions are provided in Table B.1 in Appendix B.

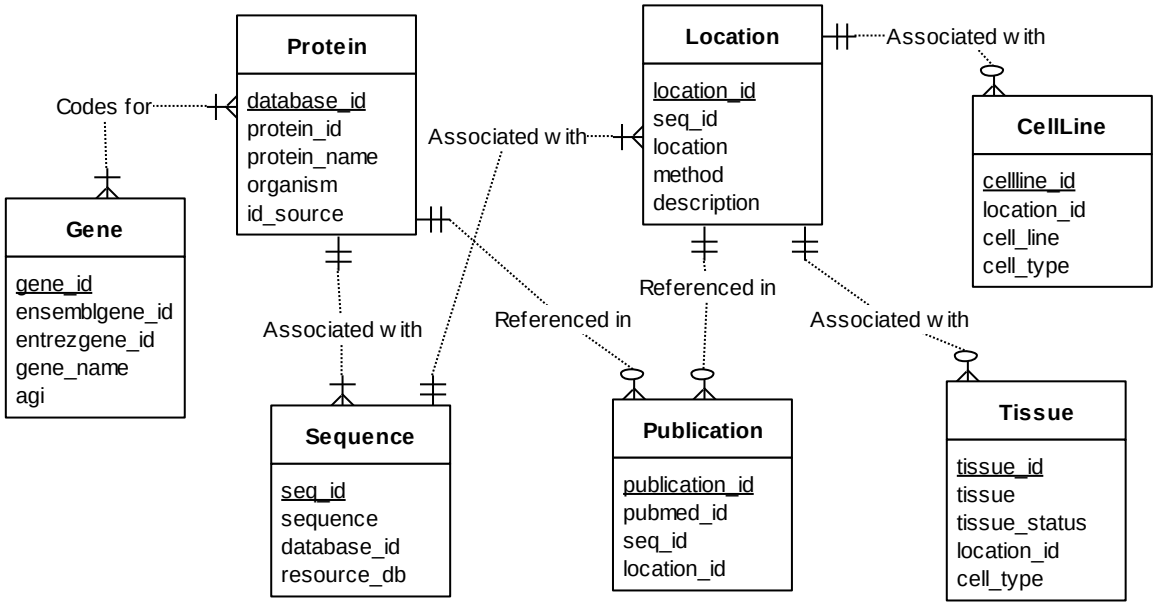


Figure 6.1: Entity relationship diagram for the protein database. Entities are represented as rectangles. Relationships between entities are shown as dotted lines. The relationship cardinality is shown by combinations of a single slash and a crow's-feet. A single slash and a crow's-feet together indicate a *one-to-many* relationship, and two crow's-feet indicate a many-to-many relationship. An optional relationship is shown using an oval.

Data Loading

We implemented the relational schema using MySQL. The implementation included the development of Python programs to create an empty database with tables defined according to the schema presented in Appendix B, to aggregate protein information extracted from various data sources in a unified format, and to insert values for protein attributes into the tables. While proteins extracted from external sources initially localized to several hundreds of locations, we focused on subcellular locations found in cells of multicellular eukaryotic organisms. Moreover, as location names were not consistent across all the sources, we standardized by mapping the names to the following *12* main subcellular compartments: *cell wall*, *chloroplast*, *cytoplasm*, *endoplasmic reticulum*, *extracellular*, *golgi apparatus*, *lysosome*, *mitochondrion*, *nucleus*, *peroxisome*, *plasma membrane*, and *vacuole*. Note that *cell wall*, *chloroplast*, and *vacuole* are only relevant to plant cells. We mapped location names that did not correspond to these 12 compartments to more general categories such as *membrane bound* and *non-membrane bound* organelles. For example, plastids such as chromoplasts and amyloplasts, cytoplasmic vesicles, and acrosome are enclosed by a membrane and fall into the former membrane-bound category. On the other hand, ribosomes and lipid droplets are particles present in the cytoplasm and referred to as non-membrane bound organlles. The mapping from location names to standardized compartments is given in Table C.1 in Appendix C. Furthermore, a summary of the information about multi-localized proteins aggregated from online sources is given in Table 6.2. This table provides for each data source, the number of sequences associated with two, three, and four or more locations.

To obtain non-redundant protein sequences, we grouped sequences that are almost identical to each other and selected one *representative* sequence per group; the rest were discarded. Specifically, we used the UCLUST algorithm [26] to efficiently cluster tens of thousands of protein sequences. In this approach, each sequence in the input set is considered as a *non-redundant* sequence and the *centroid* of a new cluster if it does not share above a certain identity that is input (different identity thresholds

Table 6.2: Summary of the protein sequences stored in our database.

Data Sources	**Number of Protein Sequences**		
	Associated with two locations	**Associated with three locations**	**Associated with four or more locations**
SUBA3	2,250	800	492
HPRD	4,927	1,849	867
HPA	4,122	1,370	209
LOCATE (Mouse)	4,735	702	104
LOCATE (Human)	5,736	1,550	611
Uniprot-KB/Swiss-Prot	32,691	5,096	1,704
Our Database	54,461	11,367	3,987

were tried as mentioned below) with any of the centroids of existing clusters; otherwise, the sequence is assigned to a cluster with the most similar centroid. Similar clustering methods have been previously used to exclude sequences/sub-fragments that are similar to each other or sequences with minor variations [98]. We note that the higher the identity threshold, lower is the number of sequences within a cluster and thus higher is the number of clusters. Additionally, employing a high identity threshold results in cluster centroids containing sequences of splice variants, whereas using a low identity threshold outputs centroids relating to sequences of proteins that are structurally different [98]. In our analysis, we tried high (i.e. 80%, 70%) as well as relatively low (i.e. 50%) identity cut-offs. Table 6.3 provides information about the *non-redundant* sequences obtained after executing the sequence elimination pipeline using different identity thresholds (i.e. 50%, 70% and 80%) on our protein collection and on the DBMLoc-derived set. We decided to use an 80% identity threshold so that the cluster centroids correspond to non-redundant sequences of both unique proteins and protein isoforms found in different organisms. The resulting cluster centroid sequences were stored in our database.

Comparison between our database and DBMLoc

The number of protein sequences associated with multiple locations in our database (*54,461*) is significantly higher than that in the DBMLoc-derived set [121] (*3,056*). Moreover, Table 6.3 shows that our database contains significantly more *dissimilar* sequences associated with multi-locations than DBMLoc. We note that a large fraction of the DBMLoc sequences (*1,761*) is absent in our database. Many such sequences are missing in our collection as they are associated with locations that are not experimentally verified. For example, for *Paralemmin-1*, a human protein that plays a role in cell formation [36], DBMLoc assigns the multi-locations, *cytoplasm* and *membrane*, based entirely on locations of other interacting proteins. Additionally, locations of about one-third of the DBMLoc protein sequences that are present in the UniProtKB/Swiss-Prot [9] repository we downloaded are outdated as they are no longer

Table 6.3: Summary of *non-redundant* protein sequences present in our protein collection and those stored in the DBMLoc-derived set.

Data Sources	**Identity Threshold**	**Number of Protein Sequences**		
		Associated with two locations	**Associated with three locations**	**Associated with four or more locations**
Our Collection	80%	26,272	4,878	1,371
DBMLoc		2,334	0	0
Our Collection	70%	22,743	4,318	1,185
DBMLoc		2,257	0	0
Our Collection	50%	17,642	3,537	936
DBMLoc		2,088	0	0

present in Swiss-Prot. Thus, DBMLoc sequences with outdated locations are not included in our database. Furthermore, for some proteins, our database stores sequences homologous to those present in DBMLoc. As another example, the sequence stored for *Nuclear mitotic apparatus protein 1* in our database is almost identical (sequence identity=99%) to that stored in DBMLoc for the same protein.

Figure 6.2 shows a comparison between the organism-wise distribution of proteins in our database and that of proteins in the DBMLoc-derived dataset. DBMLoc set predominantly contains yeast and mouse proteins. On the other hand, in our database, proteins from organisms such as yeast, mouse, Arabidopsis, and human are more uniformly distributed. Moreover, our database stores a more comprehensive collection of proteins from other eukaryotic organisms (e.g. plants, birds, fish) than DBMLoc, since our protein database was populated using several extensive online repositories.

Thus, the database of proteins that we have constructed is the most comprehensive resource of proteins localizing to *more than two* locations. We anticipate that this protein collection will help researchers to study protein movement across subcellular compartments for better understanding the functions of proteins, and to substantially evaluate the performance of current multi-location prediction systems.

In the next chapter, we present a summary of the thesis contributions and outline directions for future research.

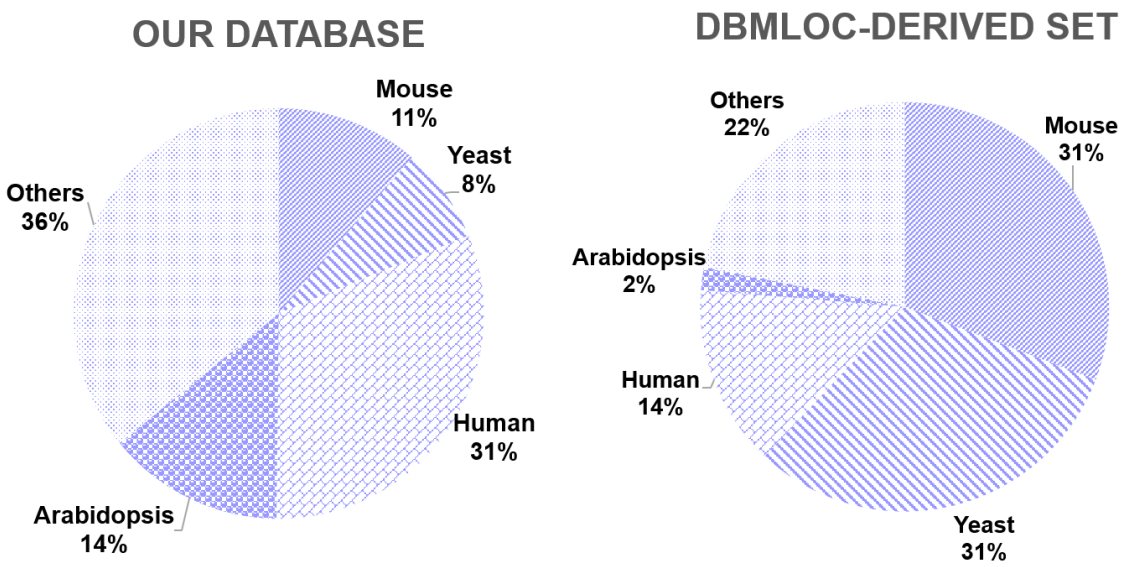


Figure 6.2: Comparison between organism-wise distribution of proteins in our database and that of proteins in the DBMLoc-derived dataset.

Chapter 7

CONCLUSIONS

This chapter summarizes the contributions made throughout the thesis and outlines directions for future research. Section 7.1 presents a summary of the machine learning methods that we introduced for protein multi-location prediction as well as for multi-label classification. The section also provides an overview of the protein database that we constructed. Section 7.2 discusses challenges posed by the introduced methods, and identifies directions for future work.

7.1 Thesis Summary and Contributions

In this work, we addressed the problem of protein multi-location prediction. While several location prediction systems already exist, most assume that proteins localize to a single location only. A few recent systems have attempted to predict multiple locations of proteins. Such systems typically treat locations as independent, even though proteins localize to possibly multiple inter-dependent locations. The goal of our research is to utilize location inter-dependencies in methods to achieve improved multi-location prediction performance.

Assigning multiple locations to proteins is a special case of multi-label classification (MLC), where proteins are the *instances* and locations are *labels*. The general problem of MLC is concerned with all classification tasks that assign multiple labels to instances. It is also our goal to apply the methods we develop to perform a number of MLC tasks and demonstrate the widespread impact of our work well beyond protein location prediction.

We addressed our goals by developing new machine learning based methods. Improved performance of these methods was shown by carrying out experiments in the

context of protein multi-location prediction as well as in the general context of MLC. The main contributions of the work are:

1. A first system for protein multi-location that improves prediction performance by using location inter-dependencies [89, 90]. We presented an initial system that incorporates inter-dependencies among locations into the prediction process. The system was based on a collection of Bayesian network classifiers, where each classifier was used to assign a single location per protein. Learning the structure of each Bayesian network classifier took into account inter-dependencies among locations, and the prediction process utilized these dependencies. Our system did not model the intricate dependencies and independencies between features and locations.

We assessed the performance of our initial system on a dataset of single- and multi-localized proteins. This dataset includes the most comprehensive set of multi-localized proteins currently available, which is derived from the DBMLoc dataset. We showed that the results obtained by using our system which employs inter-dependencies significantly improves upon those obtained by SVM classifiers which do not use any inter-dependencies. The performance of our system on multi-localized proteins was comparable to that of a top system (YLoc^{+}), but did not improve upon its performance.

2. An advanced system for multi-label classification that achieves improved performance over state-of-the-art prediction systems [91, 92, 94, 95]. We presented a probabilistic generative model that captures dependencies between features and sets of labels, in addition to representing inter-dependencies among labels as was done by our initial classifier. We introduced the concept of label dependency sets as a basis for a mixture model that explicitly represents dependencies between feature values and subsets of labels. Unlike the initial system, our advanced method used an iterative process to learn inter-dependencies among label estimates as well as among actual labels. In each iteration, the dependency structure was modified to reflect dependencies among the most recently inferred label values.

By employing the generative model to assign multiple locations to proteins, we improved upon the prediction results obtained by our initial simpler classifier as

well as by other top systems. The performance evaluation was carried out using the same comprehensive set of multi-localized proteins used in the assessment of our initial classifier. We also applied our model to a number of other MLC tasks in domains such as scene identification and song classification. We showed using experimental results that the performance of our system significantly improves upon results obtained by other current MLC systems reported in a previously published comprehensive study.

3. An extensive collection of proteins localizing to multiple locations [93]. The dataset of proteins that we used for assessing the performance of multi-location prediction systems comprises the most comprehensive current set of multi-localized proteins derived from DBMLoc. However, this set is limited to proteins that localize to only *two* locations. We thus extracted proteins specifically from online repositories that store experimentally verified information that is up-to-date, and constructed a more extensive set of multi-localized proteins. This new set contains *15,354* protein sequences (*6,279 non-redundant* sequences) associated with *three or more* locations. We designed a database for storing and efficiently accessing the protein information.

7.2 Future Work

This thesis work was concerned primarily with protein multi-location prediction. While there are several systems that assign single/multiple locations to proteins, few methods attempt to combine location assignments obtained by a collection of *similar* classifiers (e.g. BaCelLo [71]). Moreover, currently there is no method to combine assignments from many *different* state-of-the-art systems. A direction for future work is to develop a unified approach that utilizes a weighted combination of location assignments from current systems to output more accurate assignments. The weights may reflect the quality of the assignments.

Experiments comparing the performance of current multi-location prediction systems typically utilize an earlier set of multi-localized proteins that was considered

the most comprehensive. From our newly constructed extensive database, we are currently extracting proteins that localize to multiple locations and that share relatively low pairwise similarity. We anticipate that such a set of proteins can be used to substantially evaluate the performance of current systems. Furthermore, we plan to develop an interface for our database that allows users to query the resource over the web using unique protein identifiers. This functionality will help researchers to study protein movement across subcellular compartments for better understanding the functions of proteins.

While the performance of the probabilistic generative mixture model that we developed improves upon current multi-location systems as well as on other multi-label classification methods, we note that there are still a number of avenues to improve our system. For example, we employed dependency sets containing at most *three* labels in this study, and as a result, our model captures relatively simple label dependency structures. Exploring more complex dependency structures can be an interesting line for future research. However, the significant increase in the number of label dependencies may lead to prohibitively large inference times.

To reduce inference time, models that utilize trees to approximate the dependency structures instead of representing all the inter-dependencies like our current model does (e.g. *latent tree models* [12, 64, 109]) can be employed. In tree models, the leaf nodes represent observed variables and the rest of the nodes represent hidden variables. Given all the hidden variables, the observed variables are independent of each other. It would be interesting to utilize simpler structures captured by these models and examine if such a design can indeed lead to a more practical approach while still addressing most of the dependencies and independencies.

Moreover, as discussed in Chapter 4, we currently use exact inference techniques [81], which makes applying our method to datasets containing a large number (i.e. 100s) of labels impractical. Thus, another direction for future research is to develop approximate methods for label inference. Several methods for approximate

inference have been proposed in the literature. These approaches can be broadly classified into search-based algorithms (see e.g. [61, 73]), model reduction methods (e.g. *the mini-buckets scheme* [20]), and stochastic sampling techniques (e.g. *variational methods* [51], *importance sampling* [118]). We anticipate that complementary aspects of these methods can be used together to devise an efficient inference technique that considers only the label combinations likely to be associated with the instances.

An important extension to our generative model is to use the Dirichlet distribution for characterizing the mixture parameter. Such a methodology can be utilized to place more emphasis on certain dependencies between labels and features in the model.

In this work, defining label dependency sets based on a Bayesian network structure has proven useful to capture intricate dependencies and independencies between features and labels. A natural line for future research is to directly learn subsets of labels that are most likely to strongly influence feature values. Employing such label subsets in a mixture model framework could be used to effectively integrate features from different sources, e.g. from text and non-text data, and to improve multi-label classification performance.

BIBLIOGRAPHY

[1] A. Alessandro, G. Corani, D. Mauá, and S. Gabaglio. An ensemble of Bayesian networks for multilabel classification. In *International Joint Conference on Artificial Intelligence*, pages 1220–1225, 2013.

[2] R. Apweiler, A. Bairoch, C. Wu, W. Barker, and B. Boeckmann et al. UniProt: the Universal Protein knowledgebase. *Nucleic Acids Research*, 32:D115–D119, 2004.

[3] T. Bakheet and A. Doig. Properties and identification of human protein drug targets. *Bioinformatics*, 25(4):451–457, 2009.

[4] H. Bannai, Y. Tamada, O. Maruyama, K. Nakai, and S. Miyano. Extensive feature detection of N-terminal protein sorting signals. *Bioinformatics*, 18:298–305, 2002.

[5] D. Bannasch, A. Mehrle, K. Glatting, R. Pepperkok, A. Poustka, and S. Wiemann. LIFEdb: a database for functional genomics experiments integrating information from external sources, and serving as a sample tracking system. *Nucleic Acids Research*, 32:D505–D508, 2004.

[6] W. Belden and C. Barlowe. Erv25p, a component of copii-coated vesicles, forms a complex with Emp24p that is required for efficient endoplasmic reticulum to golgi transport. *J Biol Chem*, 271(43):26939–46, 1996.

[7] J. Bernal, R. Luna, A. Espina, I. Lázaro, and F. Ramos-Morales et al. Human securin interacts with p53 and modulates p53-mediated transcriptional activity and apoptosis. *Nat Genet.*, 32(2):306–11, 2002.

[8] T. Blum, S. Briesemeister, and O. Kohlbacher. MultiLoc2: Integrating phylogeny and Gene Ontology terms improves subcellular protein localization prediction. *BMC Bioinformatics*, 10:274, 2009.

[9] B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[10] S. Briesemeister, J. Rahnenfuhrer, and O. Kohlbacher. Going from where to why – interpretable prediction of protein subcellular localization. *Bioinformatics*, 26 (9):1232–1238, 2010.

[11] S. Briesemeister, J. Rahnenfuhrer, and O. Kohlbacher. YLoc - An interpretable web server for predicting subcellular localization. *Nucleic Acids Research*, 38 (Web Server issue):W497–W502, 2010.

[12] M. Choi, V. Tan, A. Anandkumar, and A. Willsky. Learning latent tree graphical models. *J. Mach. Learn. Res.*, 12:1771–1812, 2011.

[13] K. Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. *Cellular and Molecular Life Sciences*, 43(3):246–255, 2001.

[14] K. Chou and H. Shen. Euk-mPLoc: A fusion classifier for large-scale eukaryotic protein subcellular location prediction by incorporating multiple sites. *Journal of Proteome Research*, 6(5):1728–1734, 2007.

[15] K. Chou and H. Shen. A new method for predicting the subcellular localization of eukaryotic proteins with both single and multiple sites: Euk-mPLoc 2.0. *PLoS ONE*, 5(4):e9931, 2010.

[16] K. Chou and H. Shen. Plant-mPLoc: A top-down strategy to augment the power for predicting plant protein subcellular localization. *PLoS One*, 5(6):e11335, 2010.

[17] K. Chou, Z. Wu, and X. Xiao. iLoc-Euk: A multi-label classifier for predicting the subcellular localization of singleplex and multiplex eukaryotic proteins. *PLoS ONE*, 6(3):e18258, 2011.

[18] K. Chou, Z. Wu, and X. Xiao. iLoc-Hum: Using the accumulation-label scale to predict subcellular locations of human proteins with both single and multiple sites. *Molecular BioSystems*, 8(2):629–641, 2012.

[19] T. Cover and J. Thomas. *Elements of Information theory*. Wiley, New Jersey, USA, 2nd edition, 2006.

[20] R. Dechter. Mini-buckets: A general scheme for generating approximations in automated reasoning. In *International Joint Conference on Artificial Intelligence*, pages 1297–1302, 1997.

[21] M. DeGroot and M. Schervish. *Probability and Statistics*. Pearson Education, New Jersey, USA, 4th edition, 2012.

[22] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *International Conference on Machine Learning*, pages 279–286, June 2010.

[23] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39 (1):1–38, 1977.

[24] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference in Machine Learning*, pages 194–202, 1995.

[25] M. Dreger. Proteome analysis at the level of subcellular structures. *European Journal of Biochemistry*, 270(4):589–99, 2003.

[26] R. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010. URL `http://drive5.com/usearch/`. UCLUST is available in USEARCH v8.1.1861 and was last accessed on June 24, 2016.

[27] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology*, 300(4):1005–1016, 2000.

[28] J. Eppig, C. Bult, J. Kadin, J. Richardson, J. Blake, and A. Anagnostopoulos et al. The Mouse Genome Database (MGD): from genes to mice—a community resource for mouse biology. *Nucleic Acids Research*, 33:D471–D475, 2005.

[29] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[30] J. Fink, R. Aturaliya, M. Davis, F. Zhang, K. Hanson, M. Teasdale, C. Kai, J. Kawai, P. Carninci, Y. Hayashizaki, and R. Teasdale. LOCATE: a mouse protein subcellular localization database. *Nucleic Acids Research*, 34:D213–D217, 2006.

[31] National Center for Biotechnology Information. TGFA transforming growth factor alpha.

[32] L. Foster, C. de Hoog, Y. Zhang, Y. Zhang, X. Xie, V. Mootha, and M. Mann. A mammalian organelle map by protein correlation profiling. *Cell*, 125(1):187–199, 2006.

[33] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[34] M. Gama-Carvalho and M. Carmo-Fonseca. The rules and roles of nucleocytoplasmic shuttling proteins. *FEBS Letters*, 498(2-3):157 – 163, 2001.

[35] A. Garg and G. Raghava. ESLpred2: improved method for predicting subcellular localization of eukaryotic proteins. *BMC Bioinformatics*, 9:503, 2008.

[36] C. Gauthier-Campbell, D. Bredt, T. Murphy, and A. El-Husseini. Regulation of dendritic branching and filopodia formation in hippocampal neurons by specific acylated protein motifs. *Mol Biol Cell*, 15(5), 2004.

[37] L. Gold, P. Eggleton, M. Sweetwyne, L. Van Duyn, and M. Greives et al. Calreticulin: non-endoplasmic reticulum functions in physiology and disease. *FASEB J*, 24(3):665–83, 2010.

[38] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *International Conference in Machine Learning*, pages 361–368, 2004.

[39] T. Guo, S. Hua, X. Ji, and Z. Sun. DBSubLoc: database of protein subcellular localization. *Nucleic Acids Research*, 32:D122–D124, 2004.

[40] A. Hamosh. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, 30:D52–D55, 2002.

[41] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, USA, 3rd edition, 2011.

[42] M. Hanson and R. Kohler. GFP imaging: Methodology and application to investigate cellular compartmentation in plants. *Journal of Experimental Botany*, 52(356):529–539, 2001.

[43] J. He, H. Gu, and W. Liu. Imbalanced multi-modal multi-label learning for subcellular localization prediction of human proteins with both single and multiple sites. *PLoS ONE*, 7(6):e37155, 2012.

[44] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[45] A. Höglund, P. Dönnes, T. Blum, H. Adolph, and O. Kohlbacher. MultiLoc: Prediction of protein subcellular localization using N-terminal targeting sequences, sequence motifs, and amino acid composition. *Bioinformatics*, 22(10):1158–1165, 2006.

[46] P. Horton, T. Obayashi, and K. Nakai. Protein subcellular localization prediction with WoLF PSORT. In *Asian Pacific Bioinformatics Conference*, pages 39–48, 2006.

[47] P. Horton, K. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, and K. Nakai. WoLF PSORT: Protein localization predictor. *Nucleic Acids Research*, 35(Web Server issue):W585–W587, 2007.

[48] W. Huang, C. Tung, S. Ho, S. Hwang, and S. Ho. Proloc-go: Utilizing informative gene ontology terms for sequence-based prediction of protein subcellular localization. *BMC Bioinformatics*, 9:80, 2008.

[49] T. Hubbard, B. Aken, K. Beal, B. Ballester, and M. Caccamo et al. Ensembl 2007. *Nucleic Acids Research*, 35:D610–D617, 2007.

[50] F. Jensen and T. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, London, UK, 2nd edition, 2007.

[51] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[52] K. Kandasamy, S. Keerthikumar, R. Goel, S. Mathivanan, N. Patankar, B. Shafreen, S. Renuse, H. Pawar, Y. Ramachandra, P. Acharya, P. Ranganathan,

R. Chaerkady, T. Prasad, and A. Pandey. Human Proteinpedia: a unified discovery resource for proteomics research. *Nucleic Acids Research*, 37:D773–D781, 2009.

[53] D. Kim, J. Franklyn, V. Smith, A. Stratford, and H. Pemberton et al. Securin induces genetic instability in colorectal cancer by inhibiting double-stranded DNA repair activity. *Carcinogenesis*, 28(3):749–759, 2007.

[54] B. King and C. Guda. ngLOC: An n-gram-based Bayesian method for estimating the subcellular proteomes of eukaryotes. *Genome Biology*, 8(5):R68, 2007.

[55] P. Lee and H. Shatkay. BNTagger: Improved tagging SNP selection using Bayesian networks. *Bioinformatics*, 22(14):e211–e219, 2006.

[56] L. Li, Y. Zhang, L. Zou, Y. Zhou, and X. Zheng. Prediction of protein subcellular multi-localization based on the general form of Chou's pseudo amino acid composition. *Protein and Peptide Letters*, 19(4):375–387, 2012.

[57] H. Lin, C. Chen, T. Sung, S. Ho, and W. Hsu. Protein subcellular localization prediction of eukaryotes using a knowledge-based approach. *BMC Bioinformatics*, 10(Suppl 15):8, 2009.

[58] X. Liu and X. Zheng. Endoplasmic reticulum and golgi localization sequences for mammalian target of rapamycin. *Mol. Biol. Cell*, 18(3):10731082, 2007.

[59] D. Maglott, J. Ostell, K. Pruitt, and T. Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research*, 33:D54–D58, 2005.

[60] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, USA, 2008.

[61] D. Maua and C. de Campos. Anytime Marginal Maximum a Posteriori Inference. In *International Conference on Machine Learning*, 2012.

[62] A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop on Text Learning*, 1999.

[63] A. Millar, C. Carrie, B. Pogson, and J. Whelan. Exploring the function-location nexus: Using multiple lines of evidence in defining the subcellular location of plant proteins. *Plant Cell*, 21(6):1625–1631, 2009.

[64] R. Mourad, C. Sinoquet, N. Zhang, T. Liu, and P. Leray. A survey on latent tree models and applications. *J. Artif. Int. Res.*, 47:157–203, 2013.

[65] R. Murphy. Communicating subcellular distributions. *Cytometry A.*, 77(7):686–92, 2010.

[66] K. Nakai and M. Kanehisa. Expert system for predicting protein localization sites in gram-negative bacteria. *Proteins*, 11(2):95–110, 1991.

[67] S. Patra. JEPSLD: A Judgmental Eukaryotic Protein Subcellular Location Database. Master's thesis, University of Delaware, May 2013.

[68] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, F. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[69] S. Peri, J. Navarro, R. Amanchy, T. Kristiansen, and C. Jonnalagadda et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Research*, 10:2363–2371, 2003.

[70] S. Peri, J. Navarro, T. Kristiansen, R. Amanchy, and V. Surendranath et al. Human protein reference database as a discovery resource for proteomics. *Nucleic Acids Research*, 32:D497–D501, 2004.

[71] A. Pierleoni, P. Martelli, P. Fariselli, and R. Casadio. Bacello: A balanced subcellular localization predictor. *Bioinformatics*, 22:408–416, 2006.

[72] M. Pohlschroder, E. Hartmann, N. Hand, K. Dilks, and A. Haddad. Diversity and evolution of protein translocation. *Annual Review of Microbiology*, 59:91–111, 2005.

[73] David Poole. The use of conflicts in searching bayesian networks. In *The Conference on Uncertainty in Artificial Intelligence*, 1993.

[74] T. Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, and S. Kumar et al. Human Protein Reference Database - 2009 update. *Nucleic Acids Research*, 37:D767–D772, 2009.

[75] K. Pruitt, T. Tatusova, G. Brown, and D. Maglott. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic Acids Research*, 40:D130–D135, 2012.

[76] S. Rea and D. James. Moving GLUT4: The biogenesis and trafficking of GLUT4 storage vesicles. *Diabetes*, 46(11):1667–1677, 1997.

[77] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

[78] S. Rey, J. Gardy, and F. Brinkman. Assessing the precision of high-throughput computational and laboratory approaches for the genome-wide identification of protein subcellular localization in bacteria. *BMC Genomics*, 6:162, 2005.

[79] T. Rubin, A. Chambers, P. Smyth, and M. Steyvers. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208, 2012.

[80] R. Russell, R. Bergeron, G. Shulman, and H. Young. Translocation of myocardial GLUT-4 and increased glucose uptake through activation of AMPK by AICAR. *American Journal of Physiology*, 277:H643–H649, 1997.

[81] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, New Jersey, USA, 3rd edition, 2010.

[82] M. Schiffer, C. Chang, and F. Stevens. The function of tryptophan residues in membrane proteins. *Protein Engineering*, 5(3):213–214, 1992.

[83] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Massachusetts, USA, 2002.

[84] R. Seal, S. Gordon, M. Lush, M. Wright, and E. Bruford. genenames.org: the HGNC resources in 2011. *Nucleic Acids Research*, 39:D514–D519, 2011.

[85] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–S252, 2001.

[86] H. Shatkay, A. Höglund, S. Brady, T. Blum, P. Dönnes, and O. Kohlbacher. SherLoc: High-accuracy prediction of protein subcellular localization by integrating text and protein sequence data. *Bioinformatics*, 23(11):1410–1417, 2007.

[87] H. Shen and K. Chou. A top-down approach to enhance the power of predicting human protein subcellular localization: Hum-mPLoc 2.0. *Analytical Biochemistry*, 394(2):269–274, 2009.

[88] H. Shen and K. Chou. Virus-mPLoc: A fusion classifier for viral protein subcellular location prediction by incorporating multiple sites. *Journal of Biomolecular Structure and Dynamics*, 28(2):175–186, 2010.

[89] R. Simha and H. Shatkay. Protein (multi-)location prediction: Using location inter-dependencies in a probabilistic framework. In *International Workshop on Algorithms in Bioinformatics*, pages 3–17, 2013.

[90] R. Simha and H. Shatkay. Protein (multi-)location prediction: using location inter-dependencies in a probabilistic framework. *Algorithms for Molecular Biology*, 9:8, 2014.

[91] R. Simha and H. Shatkay. Protein (multi-)location prediction: Using bayesian networks for location inter-dependencies, and a mixture model. In *International Conference and Exhibition of the Society for Laboratory Automation and Screening*, 2015.

[92] R. Simha and H. Shatkay. Improved multi-label classification using inter-dependence structure via a generative mixture model. In *European Conference on Artifical Intelligence*, 2016.

[93] R. Simha, M. Lugo, D. Arnold, T. Li, and H. Shatkay. JEPSLD: Judgmental Eukaryotic Protein Subcellular Location Database. (Manuscript in preparation).

[94] R. Simha, S. Briesemeister, O. Kohlbacher, and H. Shatkay. Protein (multi-) location prediction: utilizing interdependencies via a generative model. In *International Conference on Intelligent Systems for Molecular Biology*, 2015.

[95] R. Simha, S. Briesemeister, O. Kohlbacher, and H. Shatkay. Protein (multi-) location prediction: utilizing interdependencies via a generative model. *Bioinformatics*, 12:i365–i374, 2015.

[96] A. Smith, J. Yu, T. Smulders, A. Hartemink, and E. Jarvis. Computational inference of neural information flow networks. *PLoS Computational Biology*, 2 (11):e161, 2006.

[97] J. Sprenger, J. Fink, S. Karunaratne, K. Hanson, N. Hamilton, and R. Teasdale. LOCATE: a mammalian protein subcellular localization database. *Nucleic Acids Research*, 36:D230–D233, 2008.

[98] B. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10): 1282–1288, 2007.

[99] D. Swarbeck, C. Wilks, P. Lamesch, T. Berardini, M. Hemandez, H. Foerster, D. Li, T. Meyer, R. Muller, L. Ploetz, A. Radenbaugh, S. Singh, V. Swing,

C. Tissier, P. Zhang, and E. Huala. The Arabidopsis Information Resource (TAIR): gene structure and function annotation. *Nucleic Acids Research*, 36: D1009–D1014, 2007.

[100] S. Tanz, I. Castleden, C. Hooper, M. Vacher, I. Small, and A. Millar. SUBA3: a database for integrating experimentation and prediction to define the subcellular location of proteins in Arabidopsis. *Nucleic Acids Research*, 41:D1185–D1191, 2013.

[101] A. Tokarev, A. Alfonso, and N. Segev. *Overview of Intracellular Compartments and Trafficking Pathways*. Landes Bioscience, Texas, USA, 2000.

[102] M. Tomicic, D. Aasland, T. Nikolova, B. Kaina, and M. Christmann. Human three prime exonuclease TREX1 is induced by genotoxic stress and involved in protection of glioma and melanoma cells to anticancer drugs. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, 1833(8):1832 – 1843, 2013.

[103] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *International Conference on Music Information Retrieval*, pages 325–330, 2008.

[104] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[105] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685, 2010.

[106] T. Tung and D. Lee. A method to improve protein subcellular localization prediction by integrating various biological data sources. *BMC Bioinformatics*, 10 (Suppl 1):S43, 2009.

[107] M. Uhlén, E. Björling, C. Agaton, C. Szigyarto, and B. Amini et al. A human protein atlas for normal and cancer tissues based on antibody proteomics. *Molecular & Cellular Proteomics*, 4(12):1920–1932, 2005.

[108] H. Wang, M. Huang, and X. Zhu. A generative probabilistic model for multi-label classification. In *International Conference on Data Mining*, pages 628–637, 2008.

[109] Y. Wang, N. Zhang, and T. Chen. Latent tree models and approximate inference in bayesian networks. *J. Artif. Int. Res.*, 32(1):879–900, 2008.

[110] A. Warford, W. Howat, and J. McCafferty. Expression profiling by high-throughput immunohistochemistry. *J. Immunol. Methods*, 290:81–92, 2004.

[111] I. Witten, E. Frank, and M. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[112] N. Wiwatwattana and A. Kumar. Organelle DB: a cross-species database of protein localization and function. *Nucleic Acids Research*, 33:D598–D604, 2005.

[113] N. Wiwatwattana, C. Landau, J. Cope, G. Harp, and A. Kumar. Organelle DB: an updated resource of eukaryotic protein localization and function. *Nucleic Acids Research*, 35:D810–D814, 2007.

[114] Z. Wu, X. Xiao, and K. Chou. iLoc-Plant: A multi-label classifier for predicting the subcellular localization of plant proteins with both single and multiple sites. *Molecular BioSystems*, 7(12):3287–3297, 2011.

[115] Z. Wu, X. Xiao, and K. Chou. iLoc-Gpos: A multi-layer classifier for predicting the subcellular localization of singleplex and multiplex Gram-positive bacterial proteins. *Protein and Peptide Letters*, 19(1):4–14, 2012.

[116] X. Xiao, Z. Wu, and K. Chou. A multi-label classifier for predicting the subcellular localization of gram-negative bacterial proteins with both single and multiple sites. *PLoS ONE*, 6(6):e20592, 2011.

[117] X. Xiao, Z. Wu, and K. Chou. iLoc-Virus: A multi-label learning classifier for identifying the subcellular localization of virus proteins with both single and multiple sites. *Journal of Theoretical Biology*, 284(1):42–51, 2011.

[118] C. Yuan and M. Druzdzel. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*, 43: 1189–1207, 2006.

[119] J. Zaragoza, L. Sucar, E. Morales, C. Bielza, and P. Larrañaga. Bayesian chain classifiers for multidimensional classification. In *International Joint Conference on Artificial Intelligence*, pages 2192–2197, 2011.

[120] M. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *International Conference on Knowledge Discovery and Data Mining*, pages 999–1008, 2010.

[121] S. Zhang, X. Xia, J. Shen, Y. Zhou, and Z. Sun. DBMLoc: A Database of proteins with multiple subcellular localizations. *BMC Bioinformatics*, 9:127, 2008.

Appendix

Appendix A

PROGRAM SOURCE CODES

A.1 Bayesian network classifiers

Server: redtape.cis.udel.edu

Home Directory: /eecis/shatkay/Projects/rsimha/BNCs

(refer to Readme file under the home directory for the full command sequence)

A.1.1 To discretize protein multi-location dataset

COMMAND

Splits dataset into multiple folds

python discretizer/process_arff.py

PARAMETERS

Parameter values are set at the beginning of code file, process_arff.py

nfolds: specifies number of folds dataset is split into (e.g. *nfolds*=5,10); in this study, *nfolds* is set to 5

infilename: specifies name of dataset file as ⟨filename⟩.arff that contains protein features and locations in ARFF format

(e.g. discretizer/multiloc_animals_dbm_model_features.arff)

outfilename: specifies name of training/test dataset file created for each fold; each line contains features and locations of a protein separated by commas; filename format: ⟨basename⟩_fold[1-5]_[train/test];

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train and discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test)

COMMAND

Discretizes protein features wrt each location

python discretizer/disc_split/split[1-5]/disc_fold[1-5]/discretize.py

PARAMETERS

Parameter values are set at the beginning of code file, discretize.py

nlabels: specifies number of locations; in this study, *nlabels* is set to 9

filename_train: specifies name of training dataset file in the format: ⟨basename⟩_fold[1-5]_train
(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train)

filename_test: specifies name of test dataset file in the format: ⟨basename⟩_fold[1-5]_test
(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test)

outfile_train: specifies name of training dataset file with discretized features wrt a label; filename format: ⟨basename⟩_fold[1-5]_train_label[0-8]
(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train_label0)

outfile_test: specifies name of test dataset file with discretized features wrt a label; filename format: ⟨basename⟩_fold[1-5]_test_label[0-8]
(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test_label0)

A.1.2 To assign multiple locations to proteins

COMMAND

Predicts multiple locations of proteins

python classifier/split[1-5]/fold[1-5]/label[0-9]/classify_multilabel.py

PARAMETERS

Parameter values are set at the start of code file, classify_multilabel.py

fold: specifies fold number and is set to an integer between 1 and *nfolds*; e.g. if *nfolds*=5, then *fold*=1, 2, 3, 4, 5

nlabels: specifies number of locations; in this study, *nlabels* is set to 9

filename_train: specifies name of training dataset file with discretized features wrt a label; filename format: ⟨basename⟩_fold[1-5]_train_label[0-8]
(e.g. classifier/split1/fold1/label0/hogdbm_with_prosite_go_fold1_train_label0)

filename_test: specifies name of test dataset file with discretized features wrt a label; filename format: ⟨basename⟩_fold[1-5]_test_label[0-8]
(e.g. classifier/split1/fold1/label0/hogdbm_with_prosite_go_fold1_test_label0)

A.2 Generative model for protein multi-location prediction: MDLoc

Server: redtape.cis.udel.edu

Home Directory: /eecis/shatkay/Projects/rsimha/MDLoc

(refer to Readme file under the home directory for the full command sequence)

A.2.1 To discretize protein multi-location dataset

COMMAND

Splits dataset into multiple folds

python discretizer/process_arff.py

PARAMETERS

Parameter values are set at the beginning of code file, process_arff.py

nfolds: specifies number of folds dataset is split into (e.g. *nfolds*=5,10); in this study, *nfolds* is set to 5

infilename: specifies name of dataset file as ⟨filename⟩.arff that contains protein features and locations in ARFF format

(e.g. discretizer/multiloc_animals_dbm_model_features.arff)

outfilename: specifies name of training/test dataset file created for each fold; each line contains features and locations of a protein separated by commas; filename format: ⟨basename⟩_fold[1-5]_[train/test];

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train and discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test)

COMMAND

Discretizes protein features

python discretizer/disc_split/split[1-5]/disc_fold[1-5]/discretize_single.py

PARAMETERS

Parameter values are set at the beginning of code file, discretize_single.py

nlabels: specifies number of locations; in this study, *nlabels* is set to 9

filename_train: specifies name of training dataset file in the format: ⟨basename⟩_fold[1-5]_train

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train)

filename_test: specifies name of test dataset file in the format: ⟨basename⟩_fold[1-5]_test

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test)

outfile_train: specifies name of training dataset file with discretized features; filename format: ⟨basename⟩_fold[1-5]_train_single

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train_single)

outfile_test: specifies name of test dataset file with discretized features; filename format: ⟨basename⟩_fold[1-5]_test_single

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test_single)

A.2.2 To assign multiple locations to proteins

COMMAND

Predicts multiple locations of proteins

python classifier/split[1-5]/fold[1-5]/exec.new.py

PARAMETERS

Parameter values are set at the beginning of code file, exec.new.py

fold: specifies fold number and is set to an integer between 1 and *nfolds*; e.g. if *nfolds*=5, then *fold*=1, 2, 3, 4, 5

nlabels: specifies number of locations; in this study, *nlabels* is set to 9

N_ROUND_DIGITS: number of digits rounded to after the decimal point

SMOOTHING_PARAM: fractional pseudocounts added to observed counts of events to all the parameters

filename_train: specifies name of training dataset file with discretized features; filename format: ⟨basename⟩_fold[1-5]_train_single

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_train_single)

filename_test: specifies name of test dataset file with discretized features; filename format: ⟨basename⟩_fold[1-5]_test_single

(e.g. discretizer/hogdbm_with_prosite_go/split1/fold1/hogdbm_with_prosite_go_fold1_test_single)

A.3 Generative model for multi-label classification

Server: redtape.cis.udel.edu

Home Directory: /eecis/shatkay/Projects/rsimha/GenModelMLC

(refer to Readme file under the home directory for the full command sequence)

A.3.1 To discretize multi-label datasets

COMMAND

Splits dataset (e.g. emotions, scene, yeast, genbase) into multiple folds and discretizes features

python discretizer/⟨dataset_name⟩/process_arff.py

PARAMETERS

Parameter values are set at the beginning of code file, process_arff.py

nfolds: specifies number of folds dataset is split into (e.g. *nfolds*=5,10); in this study, *nfolds* is set to 10

nlabels: specifies number of labels; in this study, *nlabels* varies from 6 till 27 based on the dataset used

infilename: specifies name of dataset file as ⟨dataset_name⟩.arff that contains features and labels in ARFF format (e.g. discretizer/emotions/emotions.arff)

outfilename: specifies name of training/test dataset file created for each fold; each line contains discretized features and labels of an instance separated by commas; filename format: ⟨dataset_name⟩_fold[1-5]_[train/test];
(e.g. discretizer/emotions/files/emotions_fold1_train and
discretizer/emotions/files/emotions_fold1_test)

A.3.2 To assign multiple labels to instances

COMMAND

Predicts multiple labels of instances

python classifier/⟨dataset_name⟩/split1/fold[1-10]⟩/exec-mlc.py

PARAMETERS

Parameter values are set at the beginning of code file, exec-mlc.py

fold: specifies fold number and is set to an integer between 1 and *nfolds*; e.g. if *nfolds*=10, then *fold*=$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$

nlabels: specifies number of labels; in this study, *nlabels* varies from 6 till 27 based on the dataset used

N_ROUND_DIGITS: number of digits rounded to after the decimal point

SMOOTHING_PARAM: fractional pseudocounts added to observed counts of events to all the parameters

filename_train: specifies name of training dataset file with discretized features; filename format: ⟨dataset_name⟩_fold[1-5]_train

(e.g. classifier/emotions/split1/fold1/emotions_fold1_train)

filename_test: specifies name of test dataset file with discretized features; filename format: ⟨dataset_name⟩_fold[1-5]_test

(e.g. classifier/emotions/split1/fold1/emotions_fold1_test)

A.4 An extensive set of multi-localized proteins

Server: redtape.cis.udel.edu

Home Directory: /eecis/shatkay/Projects/rsimha/JEPSLD

(refer to Readme file under the home directory for the full command sequence)

A.4.1 To extract protein information from external sources

COMMANDS

Extract protein information from online sources (e.g. HPA, HPRD, LOCATE, Swissprot, and Suba3)

python data-extractor/HPA/ExtractData.py

python data-extractor/HPRD/ExtractData.py

python data-extractor/LOCATE/ExtractDataMouse.py

python data-extractor/LOCATE/ExtractDataHuman.py

python data-extractor/Swissprot/ExtractData.py

python data-extractor/Suba3/ExtractData.py

COMMON PARAMETERS

Parameter values are set at the beginning of each code file

infilename: specifies name of source data file that contains protein information (e.g. data-extractor/HPA/Data/proteinatlas.xml.gz)

outfile: specifies name of the output file created for an external source that contains various protein and gene relevation information in comma-delimited format; (e.g. data-extractor/HPA/Out/HPADataFile)

locToStdLoc: stores a mapping from locations to standardized subcellular compartments (e.g. data-extractor/HPA/Data/LocationConversions)

OTHER FILE-SPECIFIC PARAMETERS

Parameter values are set at the beginning of each code file

HPA:

geneNameNoSidToSequence: stores a mapping from gene names to relevant combinations of swissprot IDs and protein sequences

sidGeneNameToSequence: stores a mapping from combinations of gene names and swissprot IDs to relevant protein sequences

LOCATE:

idtypeToId: stores a mapping for each ID type (e.g Ensembl Protein ID, Ensembl Gene ID, Refseq ID, Entrez Gene ID) from respective IDs to combinations of swissprot IDs and gene names

Suba3:

idToGenename: stores a mapping from Arabidopsis IDs to gene names

A.4.2 To create MySQL database and load data

COMMANDS

Creates MySQL tables

source data-loader/JEPSLD-Table-Scripts.sql

Loads protein information into several MySQL tables

python data-loader/PopulateJEPSLD.py

A.4.3 To extract non-redundant sequences

COMMANDS

Extracts protein sequences from database

python database-analyzer/ExtractSequencesFromDB.py

Clusters similar sequences

.database-analyzer/SequenceAlign/usearch8.1.1861_i86linux32 -cluster_fast JEPSLDSeq
-id ⟨identity-threshold⟩ -centroids JEPSLDSeq-Centroids-⟨identity-threshold⟩
-uc JEPSLDSeq-Clusters-⟨identity-threshold⟩

Computes statistics of non-redundant protein sequences in database

python database-analyzer/SequenceAlign/ExtractInfoCentroids-JEPSLD.py

PARAMETER

Parameter values are set at the beginning of each code file

fname: specifies name of output file obtained by running USEARCH in the previous step (e.g. database-analyzer/SequenceAlign/JEPSLDSeq-Centroids-80)

Appendix B

RELATIONAL SCHEMA OF PROTEIN DATABASE

Table B.1: Table definitions.

Table Name	**Column Names**	**Column Constraints**
Gene	gene_id	Primary key, int, Not Null
	gene_name	Unique key, Varchar(50)
	entrezgene_id	Unique key, Varchar(10)
	ensemblgene_id	Unique key, Varchar(20)
	agi_id	Unique key, Varchar(20)
Protein	database_id	Primary key, int, Not Null
	protein_id	Unique key, Varchar(20), Not Null
	id_source	Unique key, Varchar(20), Not Null
	protein_name	Varchar(50)
	organism	Unique key, Varchar(50), Not Null
CodedBy	gene_id	Primary key, int, Not Null, Foreign key to the *Gene* table
	database_id	Primary key, int, Not Null, Foreign key to the *Protein* table
Sequence	seq_id	Primary key, int, Not Null
	sequence	Varchar(5000), Not Null
	database_id	int, Not Null, Foreign key to the *Protein* table
	resource_db	Varchar(20), Not Null

Table B.1 (continued): Table definitions.

Table Name	Column Names	Column Constraints
Location	location_id	Primary key, int, Not Null
	seq_id	Unique key, int, Not Null, Foreign key to the *Sequence* table
	location	Unique key, Varchar(50), Not Null
	method	Unique key, Varchar(50)
	description	Unique key, Varchar(50)
Tissue	tissue_id	Primary key, int, Not Null
	seq_id	Unique key, int, Not Null, Foreign key to the *Sequence* table
	tissue	Unique key, Varchar(30), Not Null
	tissue_status	Unique key, Varchar(30)
	cell_type	Unique key, Varchar(30)
CellLine	cellline_id	Primary key, int, Not Null
	location_id	Unique key, int, Not Null, Foreign key to the *Location* table
	seq_id	Unique key, int, Not Null, Foreign key to the *Sequence* table
	cell_line	Unique key, Varchar(30), Not Null
	cell_type	Unique key, Varchar(30)
Publication	publication_id	Primary key, int, Not Null
	pubmed_id	Unique key, Varchar(100)
	alt_source	Unique key, Varchar(100)
	seq_id	Unique key, int, Not Null, Foreign key to the *Sequence* table
	location_id	Unique key, int, Not Null, Foreign key to the *Location* table

Appendix C

LOCATION MAPPING

Table C.1: A mapping from locations to standardized subcellular compartments. The leftmost column shows locations of proteins extracted from external sources. The middle column provides the mapped compartment for locations. The rightmost columns lists organisms in which the compartments can be found.

Locations	Standardized Compartment	Organisms
barrier septum, cell septum, cell wall, primary cell wall	cell wall	plant, fungal
chloroplast, chloroplast envelope, chloroplast inner membrane, chloroplast intermembrane space, chloroplast membrane, chloroplast nucleoid, chloroplast outer membrane, chloroplast stroma, chloroplast thylakoid, chloroplast thylakoid lumen, chloroplast thylakoid membrane, plastid thylakoid membrane, plastoglobule	chloroplast	plant
actin cytoskeleton, actin filament, actin patch, calyx, cell cortex, cytoplasm, cytoplasmic, cytoplasmic puncta, cytoskeleton, cytoskeleton (actin filaments), cytoskeleton (cytokinetic bridge), cytoskeleton (intermediate filaments), cytoskeleton (microtubule end), cytoskeleton (microtubules), cytosol, host cytoplasm, host cytosol, intermediate filament, microtubule, microtubule cytoskeleton, microtubule organizing center, midbody, perinuclear puncta, perinuclear region, perinuclear theca, sarcoplasm, spindle, spindle pole, spindle pole body	cytoplasm	animal, plant, fungal

Table C.1 (continued): Location to standardized compartment mapping.

Locations	Standardized Compartment	Organisms
endoplasmic reticulum, endoplasmic reticulum lumen, endoplasmic reticulum membrane, endoplasmic reticulum-like, host endoplasmic reticulum, rough endoplasmic reticulum, rough endoplasmic reticulum lumen, rough endoplasmic reticulum membrane, sarcoplasmic reticulum, sarcoplasmic reticulum lumen, sarcoplasmic reticulum membrane, smooth endoplasmic reticulum membrane	endoplasmic reticulum	animal, plant, fungal
apical lamina, apoplast, axon, axoneme, basement membrane, bleb, brush border, cell projection, cell surface, cilium, cilium axoneme, cilium membrane, dendrite, dendritic spine, dendritic spine membrane, exosome, extracellular, extracellular matrix, extracellular region, extracellular space, filopodium, filopodium membrane, filopodium tip, growth cone, host extracellular space, hyaline layer, interphotoreceptor matrix, invadopodium, invadopodium membrane, kinocilium, lamellipodium, lamellipodium membrane, microvillus, microvillus membrane, phagocytic cup, photoreceptor outer segment, podosome, podosome membrane, pollen coat, ruffle, ruffle membrane, secreted, stereocilium, stereocilium membrane, surface film, uropodium	extracellular	animal, plant, fungal
cis-golgi network, cis-golgi network membrane, golgi, golgi, golgi apparatus, golgi apparatus lumen, golgi apparatus membrane, golgi cis cisterna, golgi lumen, golgi medial cisterna, golgi membrane, golgi stack, golgi stack membrane, golgi trans cisterna, golgi trans face, golgi vesicle, golgi-like, medial-golgi, trans-golgi network, trans-golgi network membrane	golgi apparatus	animal, plant, fungal

Table C.1 (continued): Location to standardized compartment mapping.

Locations	Standardized Compartment	Organisms
lysosome, lysosome lumen, lysosome membrane, lysosomes	lysosome	animal
inner mitochondrial membrane, mitochondria, mitochondrial inner membrane, mitochondrial intermembrane space, mitochondrial matrix, mitochondrial membrane, mitochondrial outer membrane, mitochondrial-like, mitochondrion, mitochondrion envelope, mitochondrion inner membrane, mitochondrion intermembrane space, mitochondrion matrix, mitochondrion membrane, mitochondrion nucleoid, mitochondrion outer membrane, outer mitochondrial membrane, photoreceptor inner segment	mitochondrion	animal, plant, fungal
cajal body, centromere, chromosome, gem, host nucleus, kinetochore, nuclear, nuclear envelope, nuclear matrix, nuclear membrane, nuclear pore complex, nuclear speck, nucleoli, nucleolus, nucleoplasm, nucleus, nucleus but not nucleoli, nucleus envelope, nucleus inner membrane, nucleus lamina, nucleus matrix, nucleus membrane, nucleus outer membrane, nucleus speckle, pml body, telomere	nucleus	animal, plant, fungal
acrosome, acrosome inner membrane, acrosome lumen, acrosome membrane, acrosome outer membrane, amyloplast, amyloplast inner membrane, amyloplast membrane, apicoplast, autophagosome, autophagosome lumen, autophagosome membrane, azurophil granule, chromaffin granule, chromaffin granule lumen, chromaffin granule membrane, chromoplast, chromoplast membrane, chromoplast stroma, clathrin-coated vesicle, clathrin-coated vesicle membrane	other membrane bound organelles	animal, plant, fungal

Table C.1 (continued): Location to standardized compartment mapping.

Locations	Standardized Compartment	Organisms
coated vesicle, copi-coated vesicle, copi-coated vesicle membrane, copii-coated vesicle, copii-coated vesicle membrane, cvt vesicle membrane, cytoplasmic granule, cytoplasmic granule lumen, cytoplasmic granule membrane, cytoplasmic membrane-bound vesicle, cytoplasmic vesicle, cytoplasmic vesicle lumen, cytoplasmic vesicle membrane, cytoplasmic vesicles, dense-core vesicle, early endosome, early endosome membrane, early endosomes, endocytic vesicle, endoplasmic reticulum-golgi intermediate compartment, endoplasmic reticulum-golgi intermediate compartment membrane, endosome, endosome lumen, endosome membrane, endosomes, er-golgi intermediate compartment, er-golgi transport vesicle, ergic, esterosome membrane, etioplast, etioplast membrane, intracellular vesicle, late endosome, late endosome membrane, late endosomes, melanosome, melanosome lumen, melanosome membrane, multivesicular body, multivesicular body membrane, organellar chromatophore thylakoid membrane, perinuclear vesicle, phagosome, phagosome membrane, phragmoplast, plastid, plastid inner membrane, plastid membrane, plastid outer membrane, plastid stroma, preautophagosomal structure, preautophagosomal structure membrane, recycling endosome, recycling endosome membrane, secretory granule, secretory vesicle, secretory vesicle lumen, secretory vesicle membrane, secretory vesicles, synaptic vesicle, synaptic vesicle membrane, synaptic vesicles, transport vesicle, unknown granules, vesicle, vesicles, zymogen granule	other membrane bound organelles (continued)	animal, plant, fungal
a band, centriolar satellite, centriole, centrosome, h zone, i band, lipid droplet, lipid particle, lipid particles, m line, myofibril, p-body, ribosome, sarcomere, z line	other non-membrane bound organelles	animal, plant, fungal

Table C.1 (continued): Location to standardized compartment mapping.

Locations	Standardized Compartment	Organisms
glyoxysome, glyoxysome membrane, peroxisomal matrix, peroxisomal membrane, peroxisome, peroxisome matrix, peroxisome membrane, peroxisomes	peroxisome	animal, plant, fungal
adherens junction, aleurone grain membrane, apical cell membrane, apical membrane, apical plasma membrane, apicolateral cell membrane, basal cell membrane, basolateral cell membrane, basolateral membrane, basolateral plasma membrane, caveola, cell junction, cell junctions, cell membrane, cell outer membrane, clathrin-coated pit, cleavage furrow, coated pit, desmosome, endomembrane system, focal adhesion, focal adhesions, gap junction, gap junctions, growth cone membrane, hemidesmosome, host cell membrane, host membrane, immunological synapse, integral to membrane, integral to plasma membrane, lateral cell membrane, membrane, membrane associated, membrane associated unknown, membrane fraction, membrane raft, myelin membrane, peribacteroid membrane, plasma membrane, plasma membrane-like, plasmamembrane, plasmamembranecpebtanz, plasmodesma, postsynaptic cell membrane, postsynaptic density, presynaptic cell membrane, sarcolemma, synapse, synaptosome, target cell membrane, tegument membrane, tight junction, unknown membrane	plasma membrane	animal, plant, fungal
aleurone grain, lytic vacuole, parasitophorous vacuole membrane, peribacteroid space, prevacuolar compartment, prevacuolar compartment membrane, protein storage vacuole, protein storage vacuole membrane, symbiosome, vacuole, vacuole lumen, vacuole membrane	vacuole	plant, fungal

Appendix D

PERMISSIONS

Chapter 3 is based on the two papers:

- Ramanuja Simha and Hagit Shatkay, "*Protein (Multi-)Location Prediction: Using Location Inter-dependencies in a Probabilistic Framework*," in Proceedings of the International Workshop on Algorithms for Bioinformatics, September 2013. See Appendix D.1 for documentation of permission to republish this paper.

- Ramanuja Simha and Hagit Shatkay, "*Protein (Multi-)Location Prediction: Using Location Inter-dependencies in a Probabilistic Framework*," in Algorithms for Molecular Biology 9(1), 2014. See Appendix D.2 for documentation supporting that the paper is available under the Creative Commons license.

Chapters 4 and 5 are based on the two papers:

- Ramanuja Simha, Sebastian Briesemeister, Oliver Kohlbacher, and Hagit Shatkay, "*Protein (Multi-)Location Prediction: Utilizing Interdependencies via a Generative Model*," in Bioinformatics 31(12), 2015 (Intelligent Systems for Molecular Biology 2015 Proceedings). See Appendix D.3 for documentation supporting that the paper is available under the Creative Commons license.

- Ramanuja Simha and Hagit Shatkay, "*Improved Multi-Label Classification using Inter-dependence Structure via a Generative Mixture Model*," to appear in the European Conference on Artifical Intelligence, August 2016. See Appendix D.4 for documentation supporting that the paper is freely available online.

D.1 Springer License

SPRINGER LICENSE
TERMS AND CONDITIONS

Jul 26, 2016

This Agreement between Ramanuja Simha ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	3916731212566
License date	Jul 26, 2016
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	Protein (Multi-)Location Prediction: Using Location Inter-dependencies in a Probabilistic Framework
Licensed Content Author	Ramanuja Simha
Licensed Content Date	Jan 1, 2013
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the editor of the new work
Order reference number	
Title of your thesis / dissertation	A PROBABILISTIC FRAMEWORK FOR PROTEIN MULTI-LOCATION PREDICTION, AND ITS APPLICABILITY TO MULTI-LABEL CLASSIFICATION
Expected completion date	Aug 2016
Estimated size(pages)	140
Requestor Location	Ramanuja Simha 219 Smith Hall University of Delaware NEWARK, DE 19716 United States Attn: Ramanuja Simha
Billing Type	Invoice
Billing Address	Ramanuja Simha 219 Smith Hall University of Delaware NEWARK, DE 19716 United States Attn: Ramanuja Simha
Total	0.00 USD
Terms and Conditions	

Introduction
The publisher for this copyrighted material is Springer. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions

established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License

With reference to your request to reuse material on which Springer controls the copyright, permission is granted for the use indicated in your enquiry under the following conditions:

- Licenses are for one-time use only with a maximum distribution equal to the number stated in your request.
- Springer material represents original material which does not carry references to other sources. If the material in question appears with a credit to another source, this permission is not valid and authorization has to be obtained from the original copyright holder.
- This permission
• is non-exclusive
• is only valid if no personal rights, trademarks, or competitive products are infringed.
• explicitly excludes the right for derivatives.
- Springer does not supply original artwork or content.
- According to the format which you have selected, the following conditions apply accordingly:
• **Print and Electronic:** This License include use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.
• **Print:** This License excludes use in electronic form.
• **Electronic:** This License only pertains to use in electronic form provided it is password protected, on intranet, or CD-Rom/DVD or E-book/E-journal. It may not be republished in electronic open access.

For any electronic use not mentioned, please contact Springer at permissions.springer@spi-global.com.

- Although Springer controls the copyright to the material and is entitled to negotiate on rights, this license is only valid subject to courtesy information to the author (address is given in the article/chapter).
- If you are an STM Signatory or your work will be published by an STM Signatory and you are requesting to reuse figures/tables/illustrations or single text extracts, permission is granted according to STM Permissions Guidelines: http://www.stm-assoc.org/permissions-guidelines/

For any electronic use not mentioned in the Guidelines, please contact Springer at permissions.springer@spi-global.com. If you request to reuse more content than stipulated in the STM Permissions Guidelines, you will be charged a permission fee for the excess content.

Permission is valid upon payment of the fee as indicated in the licensing process. If permission is granted free of charge on this occasion, that does not prejudice any rights we might have to charge for reproduction of our copyrighted material in the future.

-If your request is for reuse in a Thesis, permission is granted free of charge under the following conditions:

This license is valid for one-time use only for the purpose of defending your thesis and with a maximum of 100 extra copies in paper. If the thesis is going to be published, permission needs to be reobtained.

- includes use in an electronic form, provided it is an author-created version of the thesis on his/her own website and his/her university's repository, including UMI (according to the definition on the Sherpa website: http://www.sherpa.ac.uk/romeo/);
- is subject to courtesy information to the co-author or corresponding author.

Geographic Rights: Scope

Licenses may be exercised anywhere in the world.

Altering/Modifying Material: Not Permitted

Figures, tables, and illustrations may be altered minimally to serve your work. You may not alter or modify text in any manner. Abbreviations, additions, deletions and/or any other

alterations shall be made only with prior written authorization of the author(s).

Reservation of Rights

Springer reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction and (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

License Contingent on Payment

While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by Springer or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received by the date due, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and Springer reserves the right to take any and all action to protect its copyright in the materials.

Copyright Notice: Disclaimer

You must include the following copyright and permission notice in connection with any reproduction of the licensed material:

"Springer book/journal title, chapter/article title, volume, year of publication, page, name(s) of author(s), (original copyright notice as given in the publication in which the material was originally published) "With permission of Springer"

In case of use of a graph or illustration, the caption of the graph or illustration must be included, as it is indicated in the original publication.

Warranties: None

Springer makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

Indemnity

You hereby indemnify and agree to hold harmless Springer and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

No Transfer of License

This license is personal to you and may not be sublicensed, assigned, or transferred by you without Springer's written permission.

No Amendment Except in Writing

This license may not be amended except in a writing signed by both parties (or, in the case of Springer, by CCC on Springer's behalf).

Objection to Contrary Terms

Springer hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction

All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.

D.2 Algorithms for Molecular Biology Copyright

Algorithms for Molecular Biology

Algorithms for Molecular Biology main menu

Copyright

- Copyright on any open access article in a journal published by BioMed Central is retained by the author(s).
- Authors grant BioMed Central a license to publish the article and identify itself as the original publisher.
- Authors also grant any third party the right to use the article freely as long as its integrity is maintained and its original authors, citation details and publisher are identified.
- The Creative Commons Attribution License 4.0 formalizes these and other terms and conditions of publishing articles.

In addition to BioMed Central's copyright policy, some journals also follow an Open Data policy and the Creative Commons CC0 1.0 Public Domain Dedication waiver applies to all published data in these journals. Further information can be found on the individual journals pages.

Where an author is prevented from being the copyright holder (for instance in the case of US government employees or those of Commonwealth governments), minor variations may be required. In such cases the copyright line and license statement in individual articles will be adjusted, for example to state '© 2016 Crown copyright'. Authors requiring a variation of this type should inform BioMed Central during or immediately after submission of their article.

Exceptions to copyright policy

Our policy pages provide details concerning copyright and licensing for articles which were previously published under policies that are different from the above. For instance, occasionally BioMed Central may co-publish articles jointly with other publishers, and different licensing conditions may then apply. In all such cases, however, access to these articles is free from fees or any other access restrictions.

Information specifically regarding permissions and reprints can be found here. Please contact us if there are questions.

ALGORITHMS FOR MOLECULAR BIOLOGY

Submit a manuscript

- Editorial Board
- Sign up to article alerts
- Impact Factor: 1.439
- ISSN: 1748-7188

D.3 Bioinformatics License to Publish

Bioinformatics

Oxford Journals > Science & Mathematics > Bioinformatics > For Authors > Instructions to Authors

INSTRUCTIONS TO AUTHORS

Please note that the journal now encourages authors to complete their copyright licence to publish form online Upon receipt of accepted manuscripts at Oxford Journals authors will be invited to complete an online copyright licence to publish form.

Please note that by submitting an this article for publication you confirm that you are the corresponding/submitting author and that Oxford University Press ("OUP") may retain your email address for the purpose of communicating with you about the article. You agree to notify OUP immediately if your details change. If your article is accepted for publication OUP will contact you using the email address you have used in the registration process.

ALL ARTICLES MUST BE SUBMITTED ONLINE. Once you have prepared your manuscript according to the Instructions below please visit the online submission website.

Instructions on submitting your manuscript online, along with the compulsory Word and LaTex* templates, are available. [*If you use bibtex, please use the bibliography style named natbib.bst]. Help with inserting figures into the templates may be found at http://office.microsoft.com/en-gb/assistance/HA010547971033.aspx.

Please read these instructions carefully and follow them strictly so that the publication process is efficient and as rapid as possible. The Editors and the Editorial Office reserve the right to return submissions that are not prepared in accordance with the following instructions.

SCOPE

Bioinformatics provides a forum for the exchange of information in the fields of computational molecular biology and post-genome bioinformatics, with emphasis on the documentation of new algorithms and databases that allows the progress of bioinformatics and biomedical research in a significant manner.

Upon submission you will be asked to provisionally select one of the following categories for your manuscript:

- Genome analysis
- Sequence analysis
- Phylogenetics
- Structural bioinformatics
- Gene expression
- Genetic and population analysis
- Systems biology
- Data and text mining
- Databases and ontologies
- Bioimage informatics

Detailed Scope Guidelines are also available for the above categories.

TYPES OF MANUSCRIPT

The following types of paper may be submitted for publication. **Please note the page limits for each type of paper.**

Original Papers (up to 7 pages; this is approx. 5000 words).

Original papers that describe new research developments in computational molecular biology, for example: models, algorithms, software involving new methods, biological databases and network information services, and their impact on molecular biology or computer science. Actual biological data, as opposed to purely simulated data, must be used.

Discovery Notes(up to 4 pages, this is approx. 3000 words) Discovery notes report biologically interesting discoveries using computational techniques. Topics may include sequence motif detection, definition of new domains, structural similarities, gene structure prediction, comparative genomics, biomolecular networks and other aspects of computational molecular biology.

The results presented should present new discoveries, bringing insight to a relevant biological problem, and not be the simple extension of current knowledge. The results are expected to proceed from the specialized use of tools, methods, and databases. The inclusion of experimental results is considered very positively.

Application Notes (up to 2 pages; this is approx. 1300 words or 1000 words plus one figure) Applications Notes are short descriptions of novel software or new algorithm implementations, databases and network services (web servers, and interfaces). Software or data must be freely available to non-commercial users. Availability and Implementation must be clearly stated in the article. Authors must also ensure that the software is available for a full TWO YEARS following publication. Web services must not require mandatory registration by the user. Additional Supplementary data can be published online-only by the journal. This supplementary material should be referred to in the abstract of the Application Note. If describing software, the software should run under nearly all conditions on a wide range of machines. Web servers should not be browser specific. Application Notes must not describe trivial utilities, nor involve significant investment of time for the user to install. The name of the application should be included in the title.

Reviews (3-8 pages) Most review papers are commissioned, although the editors welcome suggestions from prospective authors who should in the first instance submit a draft or abstract/summary no longer than a page.

Letters to the Editor Bioinformatics publishes "letters to the editors" on the broad range of topics covered by the journal, including political, technical and scientific analyses of issues related to bioinformatics and computational biology. The letters can also include the discussion of papers published by the journal.

Conference Papers Bioinformatics considers proposals for publishing conference proceeding papers, as supplementary issues or as special sections of the journal. Please be sure to include the following information in your proposal:

- What is the theme of the conference and submitted papers?
- What numbers of papers are expected to be submitted and published?
- What peer review process will be put in place by the conference organisers to recommend papers for publication in the journal?
- How many delegates are expected to attend the conference?

- Assurance that the papers proposed for publication have not and will not be published elsewhere prior to publication in Bioinformatics, or afterwards without the permission of the journal's publisher.

PAGE CHARGES

The journal strongly discourages authors from exceeding the page limits. Manuscripts exceeding the recommended limits may be rejected without review - for example manuscripts that exceed the limit by 20% or more are usually returned by the Editorial Office immediately. In the event that a manuscript does exceed the page limits we make the following charges:

- Original papers: £100/$190 per excess page (over 7 published pages*)
- Discovery notes: £100/$190 per excess page (over 4 published pages*)
- Application notes: £100/$190 per excess page (over 2 published pages*)

Orders from the UK will be subject to the current UK VAT charge. For orders from elsewhere in the EU you or your institution should account for VAT by way of a reverse charge. Please provide us with your or your institution's VAT number.

*Please note that the Word and LaTex author templates used for submission to Manuscript Central (ScholarOne) are not an exact reflection of the final typeset article, and therefore cannot be used to estimate exactly how many published pages your final typeset article will be. Please use the following word count guidelines for a more accurate indication:

Original papers: up to 7 pages, this is approx. 5000 words, Discovery notes: up to 4 pages, this is approx. 3000 words, Application notes: up to 2 pages; this is approx. 1300 words or 1000 words plus one figure.

GENERAL POLICIES

Previous publication Submission of a manuscript implies that it reports unpublished work, that it is not under consideration for publication elsewhere and that, if accepted, it will not be published elsewhere in the same form, either in English or in any other language, without the consent of the publisher. Authors should provide the references of similar work that they have already published, or which is currently under consideration by another journal. If the work has previously been presented at a conference, authors should provide details in the covering letter. The journal will consider publication of work that has previously been presented as either a short abstract or poster at a conference, **but not as a full paper**. If previously published tables, illustrations or more than 200 words of text are to be included, then the copyright holder's written permission must be obtained. Include copies of any such permission letters with your paper. Please note that if considered appropriate, plagiarism checking software may be applied to your manuscript during the editorial review process.

Authors retain the right to make a preprint version of the article available on their own personal website and/or that of their employer and/or in free public servers of preprints and/or articles in their subject area.
Authors making their manuscripts available in this way must clearly note this in their submission.
Once the manuscript has been accepted the authors must acknowledge that the article has been accepted for publication as follows:
"This article has been accepted for publication in [Journal Title] ©: [year] [owner as specified on the article]Published by Oxford University Press [on behalf of xxxxxx]. All rights reserved."
Please read the full policy on self-archiving here: www.oxfordjournals.org/access_purchase/self-archiving_policyb.html

Authorship All persons designated as authors should qualify for authorship. Each author should have participated sufficiently in the work to take public responsibility for the content. Authorship credit should be based on substantial contribution to conception and design, execution, or analysis and interpretation of data. All authors should be involved in drafting the article or revising it critically for important intellectual content, and must have read and approved the final version of the manuscript.

Other than in exceptional circumstances the Journal does not allow addition or removal of author names after submission. A satisfactory explanation for any proposed changes in authorship will be required. We will also require a letter of consent from any person whose name has been removed indicating that they agree to the removal of their name from the author list. Owing to the complexity of these rules we strongly advise authors to fix the author list before submission and not to attempt to make changes later.

Conflicts of Interest At the point of submission, Bioinformatics policy requires that each author reveal any financial interests or connections, direct or indirect, or other situations that might raise the question of bias in the work reported or the conclusions, implications, or opinions stated - including pertinent commercial or other sources of funding for the individual author(s) or for the associated department(s) or organization(s), personal relationships, or direct academic competition. When considering whether you should declare a conflicting interest or connection please consider the conflict of interest test: Is there any arrangement that would embarrass you or any of your co-authors if it was to emerge after publication and you had not declared it?

As an integral part of the online submission process, Corresponding authors are required to confirm whether they or their co-authors have any conflicts of interest to declare, and to provide details of these. If the Corresponding author is unable to confirm this information on behalf of all co-authors, the authors in question will then be required to submit a completed Conflict of Interest form to the Editorial Office. It is the Corresponding author's responsibility to ensure that all authors adhere to this policy.

If the manuscript is published, Conflict of Interest information will be communicated in a statement in the published paper.

Software If the manuscript describes new software tools or the implementation of novel algorithms the software must be freely available to non-commercial users at the time of submission, and appropriate test data should be made available. Availability must be clearly stated in the article. Authors must also ensure that the software and test data is available for a full TWO YEARS following publication. The editors of Bioinformatics encourage authors to make their source code available and, if possible, to provide access through an open source license (see www.opensource.org for examples). Authors should make every effort to use URLs that will remain stable. At the minimum, authors must provide one of: webserver, source code or binary. The name of the software should be included in the title of the paper wherever possible.

Supporting Data All data on which the conclusions given in the publication are based must be publicly available. Bioinformatics fully supports the recommendations of the National Academies regarding data sharing (see Board on Life Sciences, Sharing Publication-Related Data and Materials: Responsibilities of Authorship in the Life Sciences. Available at www.nap.edu/books/0309088593/html). If the analysis is based on new datasets, authors are encouraged to submit these to appropriate public repositories. In particular, microarray data should be submitted to one of the recognized public repositories in a MIAME compliant way (see C. A. Ball et al., Submission of Microarray Data to Public Repositories. PLoS Biology, 2, e317). In any event, all data should be made available to the journal for the purpose of peer review. If your manuscript describes a three-dimensional model of a protein that has been manually built, you should deposit it in the PMDB database (http://www.caspur.it/PMDB, see also NAR 34, 306-309). The database will return a unique identifier which you can include in your manuscript, thereby allowing readers to have access to your model.

Supplementary Data Only directly relevant material should be included in the full text of manuscripts. Supporting materials and Appendices which are not essential for inclusion in the full text, but would nevertheless benefit the reader, can be published as online-only Supplementary Data. Supplementary Data should be submitted for review, in a separate file or files from the manuscript. Authors should make sure that all additional text, figures and tables are presented in a single file to minimise the number of files. Authors should ensure that the Supplementary Data is referred to in the main manuscript at an appropriate point in the text. It cannot be altered or replaced after the paper has been accepted for publication. For the purpose of long-term preservation of this information, we require supplementary material to be published and stored on our website. It is acceptable for authors to post supplementary material on their own website in addition to this, but not in place of this.

Pre-screening At present the journal accepts 30% of manuscripts that are submitted. Therefore, to increase the efficiency of the publication process, manuscripts received in the Bioinformatics office undergo a pre-screening process. Papers that are considered to be of low significance to the readership of the journal are returned without review.

The review process At least four recommended reviewers must be provided by the author at the submission stage. A manuscript will not be assigned to an Associate Editor until this information has been provided. Manuscripts that pass the pre-screening phase are sent to two or more referees, who agree to undertake the refereeing within a short period. Authors should normally carry out any revision within four weeks. Revisions that are not received within 90 days will be treated as new submissions.

Acceptance When accepted by the editors, authors may be asked to send the files of the final manuscript to the editorial office. These files are used for typesetting and should be either Word or LaTex files.

Because accepted manuscripts are published online following acceptance, it is important that the final version of the manuscript supplied by the author contains no information regarding the citation information (volume, issue, year) or a copyright line as this will mislead readers.

Licence to Publish It is a condition of publication in Bioinformatics that authors grant an exclusive licence to Oxford University Press. This ensures that requests from third parties to reproduce articles are handled efficiently and consistently and will also allow the article to be as widely disseminated as possible. As part of the licence agreement, Authors may use their own material in other publications provided that Bioinformatics is acknowledged as the original place of publication and Oxford University Press as the Publisher. Information about the New Creative Commons licence can be found here.

Author Self-Archiving/Public Access policy from May 2005 For information about this journal's policy, please visit our Author Self-Archiving policy page

Scientific Misconduct When dealing with potential cases of misconduct the journal follows the guidelines provided by the Committee on Publication Ethics (COPE).

Bioinformatics Advance Access *Bioinformatics* Advanced Access articles are initially published in their 'Accepted Manuscript' form as soon as possible post acceptance. Subsequently, a copyedited, typeset, corrected version of the 'Corrected Proof' is also published on the Advanced Access page. More information, including how to cite Advance Access papers, can be found on the Advance Access Page.

Complimentary ISCB memberships for authors *Bioinformatics* is an official journal of the ISCB and as part of our partnership with the Society we have 200 complimentary ISCB memberships to offer our authors each year. If you are the corresponding author of a Bioinformatics paper then the ISCB will be in touch after your article has been published.

OPEN ACCESS OPTION FOR AUTHORS

Bioinformatics authors have the **option**, at an additional charge, to make their paper freely available online immediately upon publication, under the *Oxford Open* initiative. After your manuscript is accepted, as part of the mandatory licence form required of all corresponding authors, you will be asked to indicate whether or not you wish to pay to have your paper made freely available immediately. If you do not select the Open Access option, your paper will be published with standard subscription-based access and you will not be charged.

Oxford Open articles are published under Creative Commons licences. Authors publishing in *Bioinformatics* can use the following Creative Commons licences for their articles:
• Creative Commons Attribution licence (CC BY)
• Creative Commons Non-Commercial licence (CC BY-NC)

Please click here for more information about the Creative Commons licences.

Charges also vary depending on the manuscript category. Please see below for details:

Original Paper/Review – optional Oxford Open charges:
Regular charge - £1850/ $3000 / €2450
Reduced Rate Developing country charge* - £925 / $1500 / €1225
Free Developing country charge * - £0 /$0 / €0

Application Note/Letter to the Editor/Message from ISCB/Discovery Note – optional Oxford Open charges:
Regular charge - £900 / $1460 / €1192
Reduced Rate Developing country charge* - £450 / $730 / €596
Free Developing country charge* - £0 / $0 / €0

*Visit our Developing Countries page for a list of qualifying countries

The above Open Access charges are in addition to any page charges and colour charges that might apply.

Orders from the UK will be subject to the current UK VAT charge. For orders from the rest of the EU, we will assume that the service is provided for business purposes, please provide a VAT number for yourself or your institution and ensure you account for your own local VAT correctly.

You can pay Open Access charges using our Author Services site. This will enable you to pay online with a credit/debit card, or request an invoice by email or post.

Third-Party Content in Open Access papers

If you will be publishing your paper under an Open Access licence but it contains material for which you do not have Open Access re-use permissions, please state this clearly by supplying the following credit line alongside the material:

Title of content
Author, Original publication, year of original publication, by permission of [rights holder]

This image/content is not covered by the terms of the Creative Commons licence of this publication. For permission to reuse, please contact the rights holder.

MANUSCRIPT PREPARATION

Papers must be clearly and concisely written in English and within the recommended length. In the interests of speed, manuscripts are not extensively copyedited and authors are requested to check their texts carefully before submitting them so that proofs will require only correction of typographical errors.

How to prepare text and figures

For guidelines on the types of documents that can be uploaded to the online submission system, please click here.

Prepare your figures at publication quality resolution, using applications capable of generating high-resolution .tif files (1200 d.p.i. for line drawings and 350 d.p.i. for colour and half-tone artwork). The printing process requires your figures to be in this format if your paper is accepted and printed. For useful information on preparing your figures for publication, please click here. For online submission, please also prepare a second version of your figures at low-resolution for use in the review process; these versions of the figures can be saved in .jpg, .gif, .tif or .eps format. For INITIAL submission, it is preferable that you insert the low-resolution versions of the figures and tables into the word processing but you can also upload these versions as separate files.

Sections of the manuscript

Please subdivide manuscripts into the following sequence of sections, according to the type of paper:

- **Original papers:** Title page, Structured Abstract, Introduction, System and methods, Algorithm, Implementation, Discussion, References.
- **Reviews:** May be in a format best suited to subject matter, but should include Title page, Structured Abstract, Text, References. For clarity the main body of text should be sub-divided into sections.
- **Applications notes:** Title page, Short Structured Abstract, Text.
- **Discovery notes:** The description of the analysis can be up to four pages long including one or two figures. Please include an abstract. Sequences must be freely available in the database and the results of the analyses should not have been published elsewhere.

Title page

D.4 ECAI Free Online Proceedings

- EurAI Travel Grants
- World Forum
- The Hague
- Getting there…

Call for papers

SECOND CALL FOR PAPERS: ECAI 2016
22nd European Conference on Artificial Intelligence – ECAI 2016
29 August – 2 September 2016, The Hague, The Netherlands

Submit a Paper!

The biennial European Conference on Artificial Intelligence (ECAI) is Europe's premier venue for presenting scientific results in AI. Under the general theme of "AI for human values", the 22nd edition of ECAI will be held in the city of The Hague, in sight the Peace Palace, seat of the International Court of Justice, a location highly appropriate to the conference theme. The conference dates are 29 August – 2 September 2016, with the workshops taking place on 29-30 August.

We invite the submission of papers for the technical programme of the 2016 European Conference on Artificial Intelligence (ECAI 2016). High-quality original submissions are welcome from all areas of Artificial Intelligence. The following list of topics is indicative; other topics are welcome.

- Autonomous Agents and Multi-agent Systems
- Constraints, Satisfiability, and Search
- Knowledge Representation, Reasoning, and Logic
- Machine Learning and Data Mining
- Natural Language Processing
- Planning and Scheduling
- Robotics, Perception and Vision
- Uncertainty in AI
- Web and Knowledge-based Information Systems
- Cognitive Modeling and Cognitive Architectures
- Agent-based and integrated systems
- Multidisciplinary Topics

ECAI 2016 also welcomes contributions to a special topic: Artificial Intelligence for Human Values, addressing the ethical, legal, and societal impact of AI.

Important dates

- Paper submission: 15th April 2016 – 23:59:59 GMT-12 (UTC-12)
- Author rebuttal phase: May 24th-25th May 2016
- Notification of acceptance/rejection: 7th June 2016

Formatting

Submitted papers must be formatted according to the camera-ready style for ECAI'16, and submitted electronically in PDF format through ecai2016.confmaster.net. Authorship is not anonymous.

Papers are allowed eight (8) pages. An additional page containing the list of references is allowed—as long as this ninth page contains only references. Short papers, not exceeding two (2) pages, may be submitted for poster presentation and will be subjected to light review. Over-length submissions will be rejected without review.

Originality

Submissions must be original, and in particular should not previously have been formally published, accepted for publication, or be currently under review. Also, submissions must not be submitted elsewhere during the ECAI 2016 reviewing phase.

These restrictions apply to journals and conferences, not to workshops, arXiv.org repository, and similar specialized venues with a limited audience or without archival proceedings.

Submitting and reviewing

All submissions will be subject to peer review by the ECAI 2016 Program Committee, and evaluated on the basis of:

- significance of contribution
- originality of contribution
- relevance to ECAI
- technical quality
- presentation quality
- scholarship

Short paper submissions will be evaluated mostly on the basis of originality and potential impact.

2016 is a uniquely busy year for the AI reviewing community, because of the occurrence of IJCAI, ECAI and AAAI, all within months of each other. In order to ensure continuing high reviewing standards, ECAI will introduce two innovations into the reviewing process this year.

First, ECAI 2016 provides a way to improve the continuity of the reviewing process. In cases where papers were previously rejected by IJCAI2016, and have been revised for submission to ECAI, the ECAI process will enable authors to upload the IJCAI reviews of their papers and a response letter explaining changes in the revised submitted version, for consideration during the ECAI reviewing process. This innovation is intended to reduce the "lottery" effect which can happen when revised versions are reviewed by a new team of reviewers. Instructions on how to submit you IJCAI reviews and response can be found here.

Second, Senior Programme Committee members will have the power to Summarily Reject a paper if, according to the member's expert judgment, the paper falls too far below ECAI standards to justify further reviewing effort. This change to the reviewing procedure is intended to reduce the load on the reviewers.

These innovations are being supported by the confmaster system, so authors should follow confmaster instructions on submission of their papers.

Proceedings and presentation

The proceedings of the conference will be published by IOS Press, and made freely available on the IOS website. The chairs of ECAI 2016 are currently in negotiation with a well known journal to seed a special issue of ECAI papers. The details of this arrangement will be confirmed shortly.

The authors will be responsible for producing cameraready copies of papers in PDF format, conforming to the ECAI'16 formatting guidelines for inclusion in the published proceedings of the conference. At least one author of each accepted paper is required to attend the conference to present the contribution.

Workshops and events

Separate calls are issued for workshop and tutorial proposals, as well as for contributions to PAIS 2016, the Prestigious Applications of Intelligent Systems conference, and STAIRS 2016, the Starting AI Researcher Symposium.

This year ECAI will host AIckathon, the first hackathon for development of AI apps. Details of this competition, and rules for participation, will be issued separately.

Recent Posts

- Join AIckathon
- Join the ECAI Beach party!
- Program online
- Registration reminder

Collocated Events