Scientific Programming 16 (2008) 5–29
DOI 10.3233/SPR-2008-0241
IOS Press



# Semi-supervised learning for classification of protein sequence data

Brian R. King [a,*] and Chittibabu Guda [b]

[a] *Department of Computer Science, University at Albany, SUNY, Albany, NY, USA*
[b] *Gen*NY*Sis Center for Excellence in Cancer Genomics, Department of Epidemiology and Biostatistics, University at Albany, State University of New York, Albany, NY, USA*

**Abstract.** Protein sequence data continue to become available at an exponential rate. Annotation of functional and structural attributes of these data lags far behind, with only a small fraction of the data understood and labeled by experimental methods. Classification methods that are based on semi-supervised learning can increase the overall accuracy of classifying partly labeled data in many domains, but very few methods exist that have shown their effect on protein sequence classification. We show how proven methods from text classification can be applied to protein sequence data, as we consider both existing and novel extensions to the basic methods, and demonstrate restrictions and differences that must be considered. We demonstrate comparative results against the transductive support vector machine, and show superior results on the most difficult classification problems. Our results show that large repositories of unlabeled protein sequence data can indeed be used to improve predictive performance, particularly in situations where there are fewer labeled protein sequences available, and/or the data are highly unbalanced in nature.

Keywords: Bioinformatics, protein sequence classification, semi-supervised learning, expectation maximization, EM

## 1. Introduction

Proteins are the end products of gene expression, thus there is significant interest in understanding the gene function at the proteome level among the biological and medical research community. The proteome, defined as the entire set of proteins expressed in a species, has been the focus of biomedical research in recent years, attributed largely to the need to understand the functional roles of proteins in cellular metabolism that cannot be inferred solely from information derived from the genome. Recent genome sequencing projects have identified an enormous number of potential protein-coding regions (a.k.a. open reading frames or ORFs), which are computationally translated to obtain hypothetical protein sequences. These sequences are eventually characterized by experimental methods to fully annotate their functional features. Unfortunately, the rate at which proteins are being fully annotated lags far behind, resulting in large repositories of protein data whose characteristics are not known. These reasons clearly establish the critical need for computational methods that can accurately annotate the functional role of proteins on large-scale protein datasets by using the limited amount of experimentally determined labeled data [1].

Computational methods for protein classification aim to infer a model (i.e. a function) that map proteins to a set of categories, with the intent of using these models to annotate some functional attributes of unannotated proteins. Proteins, being the functional workhorse of all living entities, have a wide array of possible descriptors used to aid in organizing and classifying them, most of which are covered by the well-known Gene Ontology project [2]. Despite the large amount of possible classifications, almost all computational methods for protein classification focus on a narrow range of possible labels, in order to reduce the problem into a tractable one. By definition, proteins are linear chains of amino acids joined by peptide bonds. The information associated with the functional, structural and evolutionary characteristics of the protein is essentially contained in this sequence. The aim of all protein classification methods is to infer this information using knowledge learned from experimentally determined sets of proteins in the most accurate manner possible.

*Corresponding author: B.R. King, Department of Computer Science, University at Albany, State University of New York, Albany, NY 12222, USA. E-mail: bking@cs.albany.edu.

Supervised methods for classification induce predictive models from labeled training data. Each instance in the training data is represented as a vector of feature values that represent the instance, and a categorical assignment of the instance determined prior to model inference. Ideally, these methods aim to induce a model that can generalize over the entire domain space. This paradigm is no exception in supervised protein classification, where a model is inferred through analysis of training data consisting of proteins labeled with high-level, experimentally-determined annotation. The majority of publicly available protein data contain instances, represented by strings of letters, which identify the chain of amino acids that make up the protein. The string is determined over an alphabet of 20 characters, where each character represents a unique amino acid – a representation commonly known as the primary structure of a protein. There is rarely a problem finding protein data in their primary structure form, due to the exponential rate that these data are made available in public repositories, which far exceeds the rate of any other structural representation of proteins available. Fortunately, supervised learning methods for protein sequence classification are currently being researched and developed in earnest. However, methods to experimentally determine the function and/or structure of a protein are costly and time-consuming, resulting in large repositories of protein sequences with unknown function (i.e., unlabeled data). There are a considerable lack of methods that can make use of the comparatively large amount of unlabeled sequence data. Thus, this represents an ideal domain to explore *semi-supervised learning* methods, which are employed in situations where there are both labeled and unlabeled data available, and the amount of unlabeled data often excessively exceeds the amount of labeled data available [3].

In this paper, we consider how semi-supervised learning can be used to induce a better protein classification model than supervised learning, primarily in situations where the target classes of interest have a relatively small amount of labeled protein data, and/or there is low sequence similarity among labeled protein sequences. The primary set of target labels for empirical analysis will be drawn from the set of the most common subcellular localizations that a protein can localize into in order to perform its function. Our work presented is founded on our prior work completed on a supervised classification method for subcellular localization, called ngLOC [4]. The semi-supervised extensions applied are founded on work completed by Nigam and colleages, where semi-supervised learning methods based on expectation maximization (EM) techniques were applied to the problem of document classification [5,6]. For comparative analysis, we present empirical results from application of transductive support vector machines (TSVMs), which also apply unlabeled data for improved model inference [7]. For our work, we adopt a dataset of proteins labeled with subcellular localization, though this work can be extended to any protein classification problem where entire sequences are labeled with some type of functional annotation.

This paper is organized as follows. Section 2 will provide a brief presentation of related work in protein classification and semi-supervised learning. Section 3 will present the generative, probabilistic model that we adopt for protein classification, including the formalities of the naïve Bayes classification model applied to protein sequence classification. Section 4 will focus on the problem of learning from unlabeled protein sequence data in this context, and present the modifications to the model to incorporate unlabeled data, showing how it is theoretically possible to yield a better model than one learned from labeled data alone. We include a presentation of the EM algorithm, and present our extensions to the algorithm for our work. Section 5 presents a brief overview of the transductive SVM that we use for comparative analysis. Section 6 presents empirical results from our work, as well as comparative results from transductive SVMs. Section 7 offers a discussion regarding the practicalities of this work and discusses differences between our work and other related work.

## 2. Background and related work

There have been a plethora of methods that have appeared in recent years designed to perform various types of protein classification. The majority of methods have strived to operate solely from the protein sequence (i.e., the primary structure). This is partly due to the added computational complexity of accurately representing the folded protein, and partly due to the relatively large amount of sequence data available compared to any other structural representation. These methods aim to categorize proteins into various functional and structural classes. The primary differences among the existing protein classification methods lie in the feature space used to represent each protein into the classifier, the set of classes in which each protein is to

be classified, the algorithm and distance metrics used to learn the relationships between the features representing the protein data and the target labels, and the form of the resulting hypothesis induced that explains the relationships between the protein data and the target classes [8].

Early methods in supervised protein classification performed searches through datasets of labeled protein sequences and determined similarities by conducting sequence alignments. These methods can be broken into tools based on pairwise sequence alignments (most notable is the BLAST algorithm [9]), multiple sequence alignments [10,11], and tools based on profile hidden Markov models [12].

More recently, machine learning has offered a wide variety of supervised methods for classification. The majority of these methods are said to be inductive inference methods; that is, the method aims to derive a model (a function) from a set of training data that is then applied to new data [13]. The induced model is usually assumed to be a global model, covering all possible instances that may occur in the problem domain. From a probabilistic perspective, supervised learning is the induction of a model of the class-wise distribution over the entire sample space; i.e., a model is learned for the posterior probability $p(y|\mathbf{x})$, where $y$ represents a possible class assignment for a vector of feature values $\mathbf{x}$, which represents the instance to be classified. This model assigns the instance to the class that yields the highest posterior probability.

Classification methods are distinguished between generative classifiers, which learn a model of the joint probability distribution $p(\mathbf{x}, y)$, and discriminative classifiers, which model the posterior probability $p(y|\mathbf{x})$ directly [14]. The support vector machine (SVM) is currently among the most widely used discriminative classification method in bioinformatics, with early applications occurring in the fields of detecting remote homologs [15], subcellular localization prediction [16] and analysis of microarray gene expression data [17]. Other discriminative classifiers such as artificial neural networks and decision trees made earlier appearances in the area of secondary structure prediction [18,19]. Generative classification methods also have been extensively used in this domain, with hidden Markov models among the most prominent generative method, widely used for modeling protein families and for other structural prediction tasks [12,20]. Many methods have also adapted the naïve Bayes classification method through application of techniques used in document classification, where a protein sequence is considered to be the equivalent of a document [21–23]. There are a wide variety of methods from machine learning and data mining that have been applied for supervised protein classification; we suggest that the reader consult the vast array of literature available [8, 24].

Semi-supervised learning attempts to create a better model for classification by using both labeled and unlabeled data. Most semi-supervised learning methods are built upon existing supervised classification methods in machine learning, and aim to adjust the parameters of the underlying model to better capture information about all of the data available, including both labeled and unlabeled data. For some methods, the representation of the unlabeled data use as input to the classifier is the same as the labeled data, though quite often the model is improved through changing the representation of the data input to the classifier by incorporating the observed structure of all of the data [25–27].

The concept of semi-supervised learning has been around for quite some time; however, interest burgeoned in the research community in the 1990s due to successful application in text classification and natural language problems [3,5,7]. Though semi-supervised learning has been proven to be useful in a wide array of domains, its use in protein classification is slowly beginning to be recognized, with only a handful of methods being available. Recently, string kernels for protein sequence data were used with SVMs for classification of protein domains into superfamilies [28,29]. In this work, the string kernels were enhanced through development of scalable cluster kernel techniques that allow incorporation of unlabeled protein sequence data. In related areas of bioinformatics, semi-supervised learning was used to improve the prediction of protein function by using a graph-based representation of protein interactions. Specifically, the unlabeled data was used to improve the assignment of weights among multiple graphs representing protein interaction networks [30]. The technique also has been recently applied to improving the analysis of gene expression data resulting from microarray experiments [31].

In contrast to the inductive inference methods prevalent in supervised learning, semi-supervised learning is significantly tied to a more recent area of research known as transductive inference [32]. Classification methods based on inductive inference use a set of training data to induce a model that maps the entire sample space to a set of categories. In contrast, methods based on transductive inference focus on learning a model that can be used to classify a finite set

of test data. Though it was first introduced in the mid 1990's by Vapnik [33,34], the value and potential of transductive inference went largely unnoticed until Joachims demonstrated its utility for text classification [7]. In this work, he introduced the Transductive Support Vector Machine (TSVM), and reported substantial improvement over standard SVMs and other inductive methods. Proponents of transductive inference state that predictive performance can be improved by reducing the learning problem to one of inferring a local function to classify a specific set of data. However, this can result in a model that fails to generalize over the entire sample space, often requiring a new model to be inferred when new data are encountered. There are very few examples of transductive inference applied in bioinformatics tasks. Krogel and Scheffer [35] conducted an extensive study on the caveats of incorporating semi-supervised learning and transduction for predicting various functional properties of proteins corresponding to genes in the yeast genome. Kasabov and Pang used a transductive SVM for promoter recognition [36], improving predictive performance by 55% over the standard inductive SVM results.

We refer the reader to the excellent surveys presented in [3] and [37], both of which present much of the current research in semi-supervised learning in a wide variety of domains.

## 3. A supervised, generative model for protein classification

We have adopted a generative, probabilistic approach to protein classification, derived largely from classic Bayesian classification. This type of probabilistic classification has been used successfully in a variety of problems, including document and protein classification tasks [13,21,38,39]. Probabilistic methods have significant value in this domain because they allow the classification model to inherently represent the uncertainty in the data and the classes to which the data belong. Though modern, high-throughput proteomic methods have enhanced the pace of protein annotation, it has been delivered at a price, often yielding inconsistent, incorrect data [40,41]. Moreover, the variety of methods for experimentally determining the function and/or structure of the protein can yield inconsistent and/or inconclusive results [41]. The result is noisy data, which results in uncertainty that must be accounted for. Further uncertainty in the classifier can arise simply due to poor feature selection methods that are unable to generalize across the protein space, poor choice of non-probabilistic classification methods that cannot properly account for uncertainty in the data, or simply due to bad data resulting from error in the experimental techniques used to annotate the data. These are all common occurrences in protein sequence and annotation data, and clearly justify the need for a pure probabilistic approach to protein classification.

In our previous work, we have shown how a mixture of multinomial models in a supervised, naïve Bayes framework can be used to classify proteins over 10 distinct subcellular localizations [4]. We present formal, yet brief, analysis of this model here, in order to effectively lay the foundation required for the semi-supervised extensions to be presented.

Formally, we let $\mathcal{D}$ represent the protein space – the set of all possible proteins. The labeled data will be represented by $D_L$, and the unlabeled data will be represented by $D_U$, and $|D_L| \ll |D_U|$. $D$ is the union of $D_L$ and $D_U$. Let $\mathcal{C}$ denote the set of possible classes representing the discrete annotations that each protein may be annotated with, where $\mathcal{C} = \{c_j \mid j = 1, \ldots, |\mathcal{C}|\}$. For our work, our target classes are represented by a set of major subcellular localizations that each protein may be localized to upon synthesis (refer to Section 6). Let $\mathcal{X}$ represent the feature space – the space of features that could be used to represent any instance in $\mathcal{D}$. The function that maps instances in $\mathcal{D}$ to features over $\mathcal{X}$ is assumed to be neither injective nor surjective; that is, there may be distinct instances in $\mathcal{D}$ that map to the same feature vector, *and* there may be feature vectors over $\mathcal{X}$ for which no instance in $\mathcal{D}$ could possibly exist in nature. The only assumption is that each unique instance in $\mathcal{D}$ maps to one feature vector over $\mathcal{X}$. Let $\mathbf{x}_i$ represent the vector of feature values representing instance $d_i \in D$, and $y_i$ represent the class that $d_i$ has been assigned to. Then, $D_L = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, |D_L|\}$ and $D_U = \{\mathbf{x}_i \mid i = 1, \ldots, |D_U|\}$. We use the notation $D_{L,j}$ to represent the subset of the labeled data that belongs to class $c_j$. The notation $d_i \in D$ and $\mathbf{x}_i \in D$ are both used in this research to refer to the same $i$th instance in $D$; the representation is implied in the notation.

In general, given any protein sequence $d_i$, the aim is to develop a probabilistic model of $P(c_j|d_i)$ for each class $c_j$. The classifier $h$ will label each instance $d_i$ according to the class model resulting in the highest probability. Equation (1) shows this in probabilistic terms, and shows how the well-known Bayes rule is used to

derive an estimate for this probability:

$$\begin{aligned} h(d_i) &= \arg\max_{c_j} P(c_j|d_i) \\ &= \arg\max_{c_j} \frac{P(d_i|c_j)P(c_j)}{P(d_i)} \\ &= \arg\max_{c_j} P(d_i|c_j)P(c_j). \end{aligned} \quad (1)$$

The probability of $P(d_i)$ in the denominator is dropped, as it is independent of the class. Therefore, obtaining a model for classification involves obtaining models for the likelihood $P(d_i|c_j)$ and the prior probability $P(c_j)$.

We assume a known form for the probabilistic model prior to learning, with the parameters of the model for the likelihood denoted as $\boldsymbol{\varphi}_j$, and the parameter for the class prior denoted as $\boldsymbol{\pi}$. We let $\boldsymbol{\theta}$ denote the entire set of all parameters for the model, $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\varphi}_j \mid j = 1, \ldots, |\mathcal{C}|\}$. We assume the form of the model to be a generative, probabilistic mixture model, where a protein sequence is produced by selecting a generating component according to the prior class probabilities parameterized by $\boldsymbol{\pi}$. The selected mixture component generates a protein sequence according to its own set of parameters $\boldsymbol{\varphi}_j$ for class $c_j$. We assume a one-to-one relationship between the components in the mixture model and the classes in which data may be assigned to [38].

The probabilistic form of the model will be understood through development of the feature space we use to represent protein sequences. The feature space $\mathcal{X}$ is developed in light of the significant work that has been accomplished in the field of document classification. A study by Cheng et al. has shown that using document classification techniques on the primary sequence can achieve good results on classifying protein families [21]. In a typical document classification model, $\mathcal{X}$ is constructed by considering all possible words that may appear throughout the entire set of documents. Here, we consider subsequences of a protein of fixed-length $n$ as the equivalent of words in a document. In literature, these protein subsequences have been commonly called $n$-grams, $n$-mers, $n$-peptides, or simply words or subsequences of length $n$ [42,43]. Here, the term $n$-gram will be adopted.

Let $\Sigma$ represent the set of all possible amino acids, $|\Sigma| = 20$. Let $d_i$ be a sequence in $D$ having $k$ residues in length, where $d_i = (s_1 s_2, \ldots, s_k)$ and each $s_i \in \Sigma$. Letting $d_i^n$ be the $n$-gram representation of protein sequence $d_i \in \mathcal{D}$, we have:

$$\begin{aligned} d_i^n = \big((s_1, \ldots, s_n), (s_2, \ldots, s_{n+1}), \ldots, \\ (s_{k-n+1}, \ldots, s_k)\big), \end{aligned}$$

where each $(s_j, \ldots, s_{j+n-1})$ represents the $n$-gram in $d_i$ starting at position $j$ and continuing for $n$ residues. Overlapping subsequences are used to capture as much information as possible about the dependencies between individual residues within the entire sequence. In protein classification tasks using the $n$-gram model, $\mathcal{X}$ is constructed by considering all possible $n$-grams that may occur in proteins. In an $n$-gram model, the size of the feature space grows exponentially with $n$, as $|\mathcal{X}| = |\Sigma|^n$. Fortunately, for large values of $n$ (namely, $n > 5$), only a fraction of the theoretically possible $n$-grams actually occur in nature due to the evolutionary selection process, as a delicate mixture of various amino acid combinations are required to sustain a fold that allows the protein to carry out its designated function. Thus, we use a one-to-one mapping between unique $n$-grams and the set of integers to be used as indices. This requires memory allocation only for $n$-grams that occur in the training data, thereby allowing exploration of large values of $n$.

Let $w_t$ represent the $t$th $n$-gram in $\mathcal{X}$, and let random variable $X_t$ indicate the event that $n$-gram $w_t$ occurred $N_{it}$ times in sequence $d_i$. We assume that the probability of such an event occurrence is independent of the position in the sequence, and is independent of any other $n$-gram event occurrence (commonly called the naïve Bayes assumption). Under these assumptions, the joint distribution of $X_1, X_2, \ldots, X_{|\mathcal{X}|}$ follows a multinomial distribution for each class $c_j$, with parameters $\varphi_{1j}, \ldots, \varphi_{|\mathcal{X}|_j}$. Each class $c_j$ has its own model, and thus has its own set of parameters, denoted $\boldsymbol{\varphi}_j$ (i.e. $\boldsymbol{\varphi} = \{\boldsymbol{\varphi}_j \mid j = 1, \ldots, |\mathcal{C}|\} = \{\varphi_{tj} \mid t = 1, \ldots, |\mathcal{X}|, j = 1, \ldots, |\mathcal{C}|\}$). Each sequence $d_i$ is then represented as a vector of the number of occurrences of each $n$-gram $w_t$ that occurred in the sequence. It can be shown through an easy derivation that a sequence will be classified according to the derivation shown in Fig. 1.

### *3.1. Parameter estimation*

Let $\hat{\boldsymbol{\theta}}$ represent the estimate for the true model parameters $\boldsymbol{\theta}$. Typical naïve Bayes text classifiers determine parameter estimates through application of maximum likelihood (ML) methods or Bayesian maximum

$$
\begin{aligned}
h(d_i) &= \arg\max_{c_j} P(c_j|d_i, \boldsymbol{\theta}) \\
&= \arg\max_{c_j} P(c_j|\boldsymbol{\pi})(d_i|c_j, \boldsymbol{\varphi}_j) \\
&= \arg\max_{c_j} P(Y = c_j|\boldsymbol{\pi}) \\
&\quad \times P(\mathbf{X} = \mathbf{x}_i|Y = c_j, \boldsymbol{\varphi}_j) \\
&= \arg\max_{c_j} P(Y = c_j|\boldsymbol{\pi}) \\
&\quad \times \left( \prod_{t=1}^{|\mathcal{X}|} P(X_t = N_{it}|Y = c_j, \boldsymbol{\varphi}_j) \right) \\
&= \arg\max_{c_j} \pi_j \left( \prod_{t=1}^{|\mathcal{X}|} \varphi_{tj}^{N_{it}} \right)
\end{aligned}
$$

Fig. 1. Derivation of naïve Bayes classifier.

*a posteriori* (MAP) methods [5,13]. To estimate the class prior, we use a ML estimate (though a MAP estimate is acceptable, the difference between the two estimates is negligible due to the size of the data). The class prior estimates are calculated based on the proportion of the labeled training data belonging to that class:

$$\hat{\pi}_j = P(Y = c_j) = \frac{|D_{L,j}|}{\sum_k |D_{L,k}|}.$$

The estimate of $\hat{\varphi}_{tj}$, the probability of $n$-gram $w_t$ occurring in class $c_j$, is determined empirically through observation of actual occurrences of $w_t$ in the training data. This gives the maximized probability estimate based on the observed data. The ML estimate for probability of $n$-gram $w_t$ occurring in class $c_j$ is computed as the number of times $w_t$ occurs in class $c_j$ divided by the total number of $n$-grams in $c_j$. However, this can result in a probability of zero for some $n$-grams, as not all $n$-grams occur in all classes. The common solution that we adopt is to use a MAP estimate, which incorporates a prior distribution over the parameter space by adding a phantom occurrence of every $n$-gram in every class. This approach is commonly known as Laplace smoothing [5,44]. The estimate is given by:

$$
\begin{aligned}
\hat{\varphi}_{tj} &= \frac{1 + \sum_{d_i \in D_{L,j}} N_{it}}{|\mathcal{X}| + \sum_{d_i \in \mathcal{D}_{L,j}} (\text{Length}(d_i) - n + 1)} \\
&= \frac{1 + (\text{Count}(w_t) \text{ in class } c_j)}{|\mathcal{X}| + (\text{Total } n - \text{grams in class } c_j)}.
\end{aligned}
$$

### 3.2. Confidence score

An advantage of using a probabilistic model is that the probability of the model generating a given sequence is inherently reported for every class. While the prediction is based on the model with the highest probability, the probability score also can be used as a comparative measure against other classes. If the predicted class had a low probability, it might suggest that the second or third highest predictions also should be considered as possible classifications. If the top two classes predicted were relatively high compared to the rest of the classes, it might suggest the possibility that the sequence is localized into both locations. A probabilistic confidence score is derived for sequence $d_i$ for each possible class $c_j$, denoted *CS*($c_j|d_i$), according to the derivation presented in [4].

For a given sequence $d_i$, we define $d_{\text{null}}$ to be a sequence of null symbols of a length that is equal to the length of $d_i$. (A null symbol can be any symbol s such that $s \notin \Sigma$.) Each $n$-gram in $d_{\text{null}}$ is guaranteed to never occur in the model. The probability that each class generated $d_{\text{null}}$ (of length $k$) is given by:

$$
\begin{aligned}
&P(d_{\text{null}}|c_j) \\
&\quad = \big(1/\big(|\mathcal{X}| + (\text{Total } n \\
&\qquad - \text{grams in class } c_j)\big)\big)^{k-n+1}.
\end{aligned}
$$

Let *minNullProb* be the minimum joint probability of $d_{\text{null}}$ and class $c_j$ observed across all classes:

$$\textit{minNullProb} = \min_{c_j \in \mathcal{C}} (P(c_j)P(d_{\text{null}}|c_j)).$$

A log-odds ratio that sequence $d_i$ is targeted for location $c_j$ against *minNullProb* is calculated and then normalized by dividing by the sum over all log-odds scores, to create a separate score for each subcellular location $c_j$ for a given sequence $d_i$ as follows:

$$
\begin{aligned}
&\textit{CS}(c_j|d_i) \\
&\quad = \big(\log(P(d_i|c_j)P(c_j)) - \log(\textit{minNullProb})\big) \\
&\qquad \times \left( \sum_m \big(\log(P(d_i|c_m)P(c_m)) \right. \\
&\qquad\qquad \left. - \log(\textit{minNullProb})\big) \right)^{-1} \times 100.
\end{aligned}
$$

The range for each score will always be between 0–100, with the sum of the scores over all classes totaling 100.

## 4. Semi-supervised protein classification through EM

The expectation maximization (EM) algorithm is used in situations where ML or MAP parameter estimates of a probabilistic model are needed, yet the model consists of unobserved or latent variables [45]. The latent variables can represent missing data or missing values of existing data. In our case, the latent variables represent the missing labels in our unlabeled data $D_U$. We present a generalized EM approach applied to our problem. We follow a derivation similar to that presented in numerous resources covering EM in machine learning literature [3,5,13,46]. EM has been shown to perform well in many real-world scenarios, including various protein-related classification tasks, discovery of molecular pathways [47], mining interaction data [48], and prediction of protein families [49], among many other related tasks.

Learning can be viewed as attempting to maximize an objective function. Our objective is to find the most probable parameter estimates for our probabilistic model, given the evidence of the observed data $D = D_L \cup D_U$, formally stated as $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D)$. Using the MAP method to find the parameter estimates, we assume that a prior distribution over $\boldsymbol{\theta}$ exists, and then apply Bayes rule to this expression, yielding

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D) = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta})P(D|\boldsymbol{\theta}).$$

It would be desirable to simply maximize the function through determination of the partial derivative with respect to $\boldsymbol{\theta}$, which necessitates taking the log of both sides. However, our model contains latent variables, which makes straightforward maximization infeasible. EM solves this problem through maximization of the expected value of the latent data by finding a locally optimal set of parameters in an iterative, hill-climbing fashion.

We first derive a form for the complete joint log-likelihood of the observed and latent data as $l_c(\boldsymbol{\theta}; D) = \sum_{d_i \in D} \log P(d_i|\boldsymbol{\theta})$. We let random variable $\mathbf{X}$ represent the occurrence of the feature vector $\mathbf{x}_i$ for each given data instance $d_i$, and random variable $Y$ represents the target class. $Y$ is latent for the unlabeled data only. In semi-supervised learning, our complete log-likelihood function is defined as:

$$\begin{aligned} &l_c(\boldsymbol{\theta}; D) \\ &\quad = \log P(\boldsymbol{\theta}) + \sum_{d_i \in D_L} \log P(\mathbf{X} = \mathbf{x}_i, Y = y_i|\boldsymbol{\theta}) \\ &\qquad + \sum_{d_i \in D_U} \log \sum_{c_j \in \mathcal{C}} P(\mathbf{X} = \mathbf{x}_i, Y = c_j|\boldsymbol{\theta}). \end{aligned}$$

We need to compute the expected value of the complete log-likelihood function with respect to the latent variables. We know from elementary statistics that the expectation of a sum is equal to the sum of an expectation. Thus, for clarity, we will deal with the expectation of a single instance. To further simplify the derivation, we will assume that $Y$ is unobserved for all data. The fact that $Y$ is labeled for a small subset of our data will only improve the parameter estimates, and does not harm our derivation in any way:

$$\begin{aligned} &E_Y(l_c(\boldsymbol{\theta}; \mathbf{x}_i)) \\ &\quad = E_Y(\log P(\mathbf{X} = \mathbf{x}_i|\boldsymbol{\theta})) \\ &\quad = E_Y\left(\log \sum_{c'_j \in \mathcal{C}} P(\mathbf{X} = \mathbf{x}_i, Y = c'_j|\boldsymbol{\theta})\right) \\ &\quad = E_Y\left(\log \sum_{c'_j \in \mathcal{C}} P(Y = c'_j, \boldsymbol{\theta}) \right. \\ &\qquad \left. \times P(\mathbf{X} = \mathbf{x}_i, Y = c'_j|\boldsymbol{\theta})\right) \\ &\quad = E_Y(\log E_Y(P(\mathbf{X} = \mathbf{x}_i, Y, \boldsymbol{\theta}))) \\ &\quad \leqslant E_Y(E_Y \log(P(\mathbf{X} = \mathbf{x}_i|Y, \boldsymbol{\theta}))). \end{aligned}$$

We observe that we have a log of a sum, which makes computing any partial derivative for maximizing purposes intractable. We can apply Jensen's inequality, which gives us a lower-bound on the expectation. (This is part of the reason why local maxima are obtained for the parameter estimates during the maximization step.) More importantly, we clearly see the necessity of the iterative, divide-and-conquer approach of EM due to the recurrence of the expectation. Each E-step computes the expected values of the class assignments based on the current parameter estimates, and each M-step uses the expected values to compute better parameter estimates. This continues until the observed change in the complete log-likelihood of the data between successive iterations falls below a predetermined threshold.

**Basic-EM**($D_L, D_U, \delta$)

**Input:** $D_L$ – Set of labeled protein sequence data
$D_U$ – Set of unlabeled protein sequence data

**Output:** Classifier $h$ with parameter estimates $\hat{\boldsymbol{\theta}}$

1. Train initial naive Bayes classifier using $D_L$, deriving MAP parameters estimates to find $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D_L)$
2. $k = 0$
3. **REPEAT**
   a. **E-step** – Use classifier parameterized by $\hat{\boldsymbol{\theta}}^{(k)}$ to determine new expected class assignments $\hat{\mathbf{y}}^{(k)}$ for unlabeled data.
   b. **M-Step** – Use $D_L$, current expected class assignments $\hat{\mathbf{y}}^{(k)}$ for $D_U$ and current parameters $\hat{\boldsymbol{\theta}}^{(k)}$ to compute new parameter estimates $\hat{\boldsymbol{\theta}}^{(k+1)}$
   c. $k = k + 1$
   **UNTIL** $l_c(\hat{\boldsymbol{\theta}}^{(k)}; D, \hat{\mathbf{y}}^{(k)}) - l_c(\hat{\boldsymbol{\theta}}^{(k-1)}; D, \hat{\mathbf{y}}^{(k-1)}) < \delta$
4. Return parameter estimates

Fig. 2. Basic EM algorithm.

We introduce the notation $\hat{\mathbf{y}}$ to represent the estimated class assignments for $D$, where $\hat{\mathbf{y}}_i$ represents the expected class assignments for instance $\mathbf{x}_i$. By holding the current expected values constant, we can calculate new parameter estimates using supervised learning methods. The complete log-likelihood is restated, indicating the dependence on the estimated class assignments:

$$\begin{aligned} & l_c(\boldsymbol{\theta}; D, \hat{\mathbf{y}}) \\ & \quad = \log P(\boldsymbol{\theta}) + \sum_{d_i \in D_L} \log P(\mathbf{X} = \mathbf{x}_i, Y = y_i; \boldsymbol{\theta}) \\ & \qquad + \sum_{d_i \in D_U} \sum_{c_j \in \mathcal{C}} \log P(\mathbf{X} = \mathbf{x}_i, Y = c_j; \boldsymbol{\theta}, \hat{\mathbf{y}}_i). \end{aligned}$$

We use the notation $\hat{\mathbf{y}}_i^{(k)}$ to denote the vector of new expected class assignment estimates computed for instance $\mathbf{x}_i$ after the $k$th iteration of EM. The vector $\hat{\mathbf{y}}_i^{(k)}$ has $j = 1, \ldots, |\mathcal{C}|$ entries, where each entry represents $P(Y = c_j|\mathbf{X} = \mathbf{x}_i; \hat{\boldsymbol{\theta}})$, the estimated probability that instance $\mathbf{x}_i$ belongs to class $c_j$, given the instance and the current parameter estimates. We let $\hat{\boldsymbol{\theta}}^{(k)}$ denote the current parameter estimates after performing the $k$th iteration of EM. Our two steps precisely consist of:

**E-step:** Compute $\hat{\mathbf{y}}_i^{(k)} = (\forall c_j \in \mathcal{C}, P(Y = c_j|X = \mathbf{x}_i; \hat{\boldsymbol{\theta}}^{(k-1)})), \forall \mathbf{x}_i \in D_U$;

**M-step:** Compute $\hat{\boldsymbol{\theta}}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D; \hat{\mathbf{y}}^{(k)})$.

Notice that we compute the expected class assignments of the unlabeled data only. For the labeled data, the vector $\hat{\mathbf{y}}_i^{(k)}$ is a vector of zeros except for its labeled class entry, in which it is a one. This can only improve the estimates derived in the M-step, because we know the true class assignments for a portion of our data. Figure 2 displays the basic EM algorithm.

### *4.1. Extending the EM algorithm*

We describe three extensions applied to the basic EM algorithm that improved classification performance under most circumstances. The enhancements are designed to overcome the limitations brought about by the stated assumptions of our model, to incorporate *a priori* biological knowledge into the classifier, and to select better instances from the unlabeled data to improve predictive results.

#### *4.1.1. Weighted EM using EM-λ*

The first extension is called EM-$\lambda$ [5]. It adds a new parameter $\lambda$ to control the extent to which the expected class assignments of the unlabeled data adjusts the new parameter estimates. The $\lambda$ parameter weights the unlabeled data by a factor of $\lambda$ when computing new parameter estimates, where $(0 < \lambda \leqslant 1)$. When $\lambda = 1$, each unlabeled protein sequence will have the same effect as labeled sequences, which is the equivalent to the basic EM algorithm. The resulting complete log-likelihood is modified as fol-

lows:

$$l_c(\boldsymbol{\theta}; D, \hat{\mathbf{y}}) = \log P(\boldsymbol{\theta}) + \sum_{d_i \in D_L} \log P(\mathbf{X} = \mathbf{x}_i, Y = y_i; \boldsymbol{\theta}) + \lambda \Bigg( \sum_{d_i \in D_U} \sum_{c_j \in \mathcal{C}} \log P(\mathbf{X} = \mathbf{x}_i, Y = c_j; \boldsymbol{\theta}, \hat{\mathbf{y}}_i) \Bigg).$$

This option may be useful in situations where the amount of unlabeled data significantly dominates the labeled data to the point of skewing the reasonable initial estimates formed by the initial step in the EM algorithm.

#### *4.1.2. Multi-EM – Multiple mixture components per class*

The second enhancement we made to the basic EM algorithm is called Multi-EM, and is based on relaxing the assumption that there is a one-to-one correspondence between the components in the mixture model and the classes [5,6]. There is biological justification for relaxing this assumption, particularly when developing a model for subcellular localization. When a protein sequence is synthesized, it is targeted for one or more organelles where it will carry out its function. However, a few organelles in the cell are made of several distinct subcompartments or suborganelles. For example, the mitochrondria has an outer membrane that encloses the structure inside the cell. Below this, there is an inner membrane, which separates the matrix region of the organelle from the intermembrane space. These represent four distinct suborganelles of mitochondria, and proteins that localize to the mitochondria actually localize into one of these suborganelles. Unfortunately, most protein sequences that have subcellular localization annotated are done so only at the organelle level, due to the difficulty of experimentally determining localization of a protein at this level of granularity. However, we can use this biological knowledge and enhance our mixture model to assume that each component represents a suborganelle, and then organize the component structure in the model *a priori* to group together components at the level of the organelle. In other words, we can assume there are some number of generating components belonging to each class, which relaxes the restricted one-to-one correspondence and assumes a many-to-one relationship between components and classes.

We introduce a new latent variable, denoted $Z$, into our model for each data member, which represents the suborganelle in which each protein sequence is localized. Note that this is unobserved for both labeled and unlabeled data. Once again, we can use the EM algorithm to estimate the parameters over latent variable $Z$ using the same method as the basic EM algorithm, with the exception that the expected subclass assignments are going to be generated for $D_L$ and $D_U$ during each iteration – whereas the basic EM algorithm generated expect class assignments only for the unlabeled data $D_U$. The primary difference is in the interpretation of the probability of the target class. Because we know *a priori* the relationship between components and target classes, the probability of a given class will simply be the sum of the probabilities of each of the components belonging to that class.

Let $\mathcal{Z}$ represent the total set of components (i.e., suborganelles) in the mixture model, and $z_a \in \mathcal{Z}$ will represent one distinct component in the model. We will use the notation $\mathcal{Z}[c_j]$ to represent the subset of components that are assigned *a priori* to class $c_j$. We still use $\boldsymbol{\theta}$ to refer to the parameter space, but now $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\varphi}_a \mid a = 1, \ldots, |\mathcal{Z}|\} = \{\varphi_{ta} \mid t = 1, \ldots, |\mathcal{X}|, a = 1, \ldots, |\mathcal{Z}|\}$. Under these modifications, the probability of class $c_j$, given a sequence, is defined as the sum of the probabilities of all components that were assigned to the class. Simplifying the derivation for classification shown in Fig. 1, we assign a class to a new protein sequence as follows:

$$h(d_i) = \arg\max_{c_j} \sum_{z_a \in \mathcal{Z}[c_j]} \left( \pi_a \prod_{t=1}^{|\mathcal{X}|} \varphi_{ta}^{N_{it}} \right).$$

The modifications required to the EM algorithm and underlying model to support this extension are minor, provided that the *a priori* assignments of components to individual class assignments are maintained throughout. The algorithm is presented in Fig. 3. For the labeled data, we restrict the expected component membership assignments to only those components belonging to the class in which $d_i$ has been assigned to, and always renormalize over only these classes. We let $\hat{\mathbf{z}}$ represent the vector of expected component assignments computed during each iteration of EM, where each entry in the vector represents the probability that a given sequence belongs to that component. To calculate class membership for class $c_j$, we

**Multi-EM**$(D_L, D_U, \delta)$

**Input:** $D_L$ – Set of labeled protein sequence data
$D_U$ – Set of unlabeled protein sequence data

**Output:** Classifier $h$ with parameter estimates $\hat{\boldsymbol{\theta}}$

1. For each $d_i \in D_L$ belonging to class $c_j$, randomly assign each document to some component $z_j \in \mathcal{Z}[c_j]$.
2. Train initial naive Bayes classifier using $D_L$, deriving MAP parameters estimates to find $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D_L)$
3. $k = 0$
4. **REPEAT**
   a. **E-step** – Use classifier, parameterized by $\hat{\boldsymbol{\theta}}^{(k)}$ to determine new expected subclass assignments $\hat{\mathbf{z}}^{(k)}$ for $D_L$ and $D_U$.
   b. **FOR EACH** $d_i \in D_L$, renormalize $\hat{\mathbf{z}}^{(k)}$ over components $z_a \in \mathcal{Z}[y_i]$ and set all other entries to zero.
   c. **M-step** – Use expected class assignments $\hat{\mathbf{z}}^{(k)}$ for $D_L$ and $D_U$ and current parameters $\hat{\boldsymbol{\theta}}^{(k)}$ to compute new parameter estimates $\hat{\boldsymbol{\theta}}^{(k+1)}$
   d. $k = k + 1$
   **UNTIL** $l_c(\hat{\boldsymbol{\theta}}^{(k+1)}; D) - l_c(\hat{\boldsymbol{\theta}}^{(k+1)}; D) < \delta$
5. Return parameter estimates

Fig. 3. Algorithm for Multi-EM.

simply sum the vector entries representing the components assigned to class $c_j$. The rest of the classification and EM algorithm proceeds in the same manner.

#### *4.1.3. EM-CS – Improved selection of unlabeled data*

Our third, novel extension to the basic EM algorithm is designed to intelligently select better examples from the unlabeled dataset to use for determining better parameter estimates. A smaller set of unlabeled data will allow for improved running time of the EM algorithm. Moreover, carefully chosen examples from $D_U$ may yield an improvement in predictive performance. Most protein classifiers are designed to predict a limited number of classes over all of the possible functional annotations that could be considered. However, the reality is that the unlabeled protein data $D_U$ will likely contain sequences that are not related to any of the classes that are being predicted. For example, if we are developing a binary classifier to predict whether a sequence will localize to the nucleus or cytoplasm, our training dataset $D_L$ will contain only sequences targeted to those organelles. However, it must be expected that data in $D_U$ is going to contain sequences targeted to other organelles in the cell. It would be beneficial if we could select sequences from the unlabeled data that have a higher likelihood of being organized to one of the organelles of interest, and only include those sequences in the unlabeled data.

We implemented an extension to the basic EM algorithm called EM-CS, which uses our confidence score (*CS*) to select the proteins from the unlabeled data that achieve a *CS* score $\geqslant CSthresh$ (see Fig. 4). This subset of proteins is then further reduced by selecting the number of instances in each class to be in accordance with the current estimates for $\boldsymbol{\pi}$ – the class prior parameters. These selected proteins are then used during the M-step of the algorithm to determine better parameter estimates. Sequences are allowed to be added to the selected subset during each iteration. This will allow the improved parameter estimates determined in each E-step to iteratively expand the $n$-gram feature space. This is in contrast to the basic EM algorithm, which incorporates all sequences in $D_U$ from the very first iteration, thereby incorporating many features that are unlikely to be observed because the features are significant for data belonging to other classes. By assuming a confidence score threshold, we increase the likelihood that only sequences associated with one of the classes of interest are included in the model. If we can selectively add sequences that are related to one of the classes of interest, then it should increase the likelihood that discriminatory information about $n$-grams in the test data will be determined.

**EM-CS**($D_L, D_U, CSthresh, \delta$)

**Input:** $D_L$ – Set of labeled protein sequence data
$D_U$ – Set of unlabeled protein sequence data
$CSthresh$ – Minimum confidence score allowed on $D_U$ sequences

**Output:** Classifier $h$ with parameter estimates $\hat{\boldsymbol{\theta}}$

1. Train initial naive Bayes classifier using $D_L$, deriving MAP parameters estimates to find $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|D_L)$
   $k = 0$
3. **REPEAT**
   a. **E-step** – Use classifier, parameterized by $\hat{\boldsymbol{\theta}}^{(k)}$, to determine new expected class assignments $\hat{\mathbf{y}}^{(k)}$ for $D_U$
   b. **FOR EACH** $d_i \in D_U$, flag sequences whose confidence score $\geqslant CSthresh$. Select sequences in proportion to the current class probabilities $P(c_j|\hat{\boldsymbol{\theta}}^{(k)})$
   c. **M-step** – Use $D_L$, only flagged sequences in $D_U$ and their current expected class assignments $\hat{\mathbf{y}}^{(k)}$, and current parameters $\hat{\boldsymbol{\theta}}^{(k)}$to compute new parameter estimates $\hat{\boldsymbol{\theta}}^{(k+1)}$
   d. $k = k + 1$
   **UNTIL** $l_c(\hat{\boldsymbol{\theta}}^{(k)}; D) - l_c(\hat{\boldsymbol{\theta}}^{(k-1)}; D) < \delta$
4. Return parameter estimates

Fig. 4. EM-CS algorithm.

## 5. Transductive SVMs as a comparative method

The transductive support vector machine (TSVM) will represent the method used for comparison against the EM-based methods presented in this study. As noted previously, there are very few applications in bioinformatics that use semi-supervised and transductive inference methods. However, the inductive support vector machine (SVM) is perhaps the most widely used classification model in this field. We present a very brief introduction to the SVM and TSVM methods here to give the interested reader an understanding of the methods used for comparison purposes. We encourage the interested reader to consult the vast number of available resources on SVM and kernel methods for further information [33,34].

The support vector machine (SVM) is a discriminative, supervised classification method that aims to find a hyperplane $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ that maximally separates positive and negative instances of the input data, known as the maximum-margin hyperplane [50]. (Quite often, the data are projected into a higher dimensional space through applying the kernel trick to the input vector [33], but this is beyond the scope of this work.) The data instances that are closest to the hyperplane limit the size of the margin, and are known as the support vectors. Letting $\mathbf{x}_i$ represent the vector of feature values for instance $d_i$, and $y_i$ represent the class that $d_i$ belongs to, and $y_i \in \{-1, 1\}$, the SVM learning method involves solving an optimization problem under a specified set of constraints:

$$\begin{aligned} &\text{Minimize:} && \frac{1}{2}\|\mathbf{w}^2\| + C\left(\sum_i \xi_i\right), \\ &\text{Subject to:} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1 - \xi_i \\ & && (1 \leqslant i \leqslant |D_L|). \end{aligned} \tag{2}$$

The standard SVM algorithm is referred to as a hard-margin SVM, and requires the data to be linearly separable. Alternatively, soft margin SVMs are frequently used when data may not be linearly separable, in which case slack variables $\xi_i$ are introduced, as presented in Eq. (2). These slack variables measure the degree that instance $\mathbf{x}_i$ is misclassified and allow a maximally separating hyperplane to be found even when there are instances that are misclassified.

The transductive SVM attempts to improve the predictive performance of SVMs by using unlabeled data [7,33]. As is the case with inductive SVMs, a maximal separating hyperplane is learned from labeled data, but extra constraints are introduced to ensure that the hyperplane is maximally separating unlabeled data as well. Letting $\mathbf{x}_j^*$ denote the $j$th instance in unlabeled dataset $D_U$, the aim is now to find a labeling $y_j^*$ for

each datum $\mathbf{x}_j^*$ in the unlabeled dataset, and hyperplane parameters $\mathbf{w}$ and $b$ such the hyperplane separates the labeled and unlabeled data with maximum margin. Slack variables are introduced for the unlabeled data as they were for labeled data in (2) in the event that the unlabeled data are non-separable. The optimization is now formulated as:

$$\begin{aligned} \text{Minimize:} \quad & \frac{1}{2}\|\mathbf{w}^2\| + C\left(\sum_i \xi_i\right) \\ & + C^*\left(\sum_j \xi_j^*\right), \\ \text{Subject to:} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geqslant 1 - \xi_i \\ & (1 \leqslant i \leqslant |D_L|), \\ & y_j^*(\mathbf{w}\cdot\mathbf{x}_j^* + b) \geqslant 1 - \xi_j^* \\ & (1 \leqslant j \leqslant |D_U|). \end{aligned} \tag{3}$$

The parameters $C$ and $C^*$ are user-defined parameters set before learning commences. The $C$ parameter allows trading off margin size against misclassified data, and the $C^*$ parameter penalizes unlabeled data that is inside the margin between the support vectors.

## 6. Experimental results

In this section, we present our results showing how unlabeled data can be used to improve performance over models inferred using labeled data alone. Our tests are designed to replicate many real-world situations that occur in protein classification, including limited labeled data available and unbalanced classes. Results will indicate that the smaller the amount of labeled data observed initially, the more dramatic the effect that unlabeled data can have on improving the performance of the classifier. We use the problem of subcellular localization prediction as the platform in which to test the method. The dataset used for the various tests, the test configurations, and the measures used to critically assess the performance of the classification are all presented here.

For comparative purposes, we present the results of using a transductive SVM implementation called UniverSVM, which integrates multi-class training and prediction, incorporates cross validation in generating predictive results, and is optimized for large scale datasets [51]. We chose a linear kernel because it outperformed the other kernels on these tests. We varied the $C$ and $C^*$ parameters over a wide range of selections, and chose the best results for comparison purposes. We modified the original code to include reporting of our standard performance measures over multiple classes.

### 6.1. Datasets

We use varying subsets of possible subcellular localizations as the set of target classes that the data are to be classified as. Our datasets are configured specifically to address common challenges in datasets used for protein classification. Though we use subcellular localization as our target class labels, this work can be applied to any protein classification problem where the entire sequence is to be labeled.

For this work, we started with the same dataset that was used in our ngLOC method [4]. The set of protein sequences were taken from the Swiss-Prot database, release 50.0 (May 2006), that contains experimentally determined annotations on subcellular localizations [52]. The following filters were applied to obtain high-quality data for training and testing purposes: (i) only eukaryotic, non-plant sequences were considered, (ii) sequences with predicted or ambiguous localizations were removed, (iii) sequences shorter than 10 residues in length were removed, (iv) all redundant sequences were removed, and (v) sequences known to localize in multiple locations were manually checked and sorted to avoid errors caused by automated keyword-based sorting.

To reduce the similarity of sequences in the dataset, we used the cd-hit clustering software [53] to reduce the dataset to a maximum sequence identity of 60%. Using this clustered dataset, we created three separate datasets. Our tests sometimes use randomly selected fractions of these datasets to compare how differing amounts of labeled data affect the predictive performance of the classifier. It is understood that sequence similarity is certainly much less than 60% in these fractions selected, though the similarity was not directly measured. Our three primary dataset configurations are presented in Table 1.

Our simplest test dataset, $D_L$-2A, is a balanced, two-class subset of sequences annotated to be localized to either the extracellular space (EXT) or to the plasma membrane (PLA). These two classes are chosen because they both have close prior class distributions in the original dataset. From a biological perspective, they represent two localizations of interest in the biomedical research community, as proteins that

Table 1
Labeled datasets of proteins over subcellular localizations

| Organelle | Code | $D_L$-2A | $D_L$-2B | $D_L$-3 |
|---|---|---|---|---|
| Extracellular | EXC | 2,745 | | |
| Golgi apparatus | GOL | | | 378 |
| Lysosome | LYS | | | 80 |
| Mitochrondria | MIT | | 1,155 | |
| Nuclear | NUC | | 3,814 | |
| Plasma membrane | PLA | 2,596 | | |
| Perixosomes | POX | | | 126 |
| Total | | 5,341 | 4,969 | 584 |

are targeted to these regions are interesting for drug targeting purposes. The second dataset, $D_L$-2B, is designed to evaluate the Multi-EM algorithm. These two classes are chosen because they represent organelles in the cell that are known to have multiple suborganelles, and they have an unbalanced distribution. The third dataset, $D_L$-3, was designed to test the classification of sequences where the target classes represent a small fraction of all possible classes that the data may be annotated with. For this set, we created a three-class dataset of sequences localized into the golgi apparatus (GOL), lysosomes (LYS), and perixosomes (POX). These classes are unbalanced, and represent some of the smallest organelles in the cell. (As presented in the ngLOC method, these three classes combined represent only 2.3% of the entire original dataset [4].) Thus, the annotated data available for these organelles is relatively limited. Location-wise distribution of each of these datasets is shown in Table 1.

Our unlabeled dataset $D_U$ was taken from a set of 76,997 eukaryotic, non-plant protein sequences downloaded from the Swiss-Prot database [52]. We filtered out sequences that did not belong to eukaryotic, non-plant species prior to classification to reduce the likelihood of using data that belonged to classes outside of our domain. For example, plant cells have proteins localized to the chloroplast and vacuole, which are organelles that typically do not exist in eukaryotic animal cells. We removed all sequences in $D_U$ that already exist in the set of labeled data before beginning each test.

### 6.2. Performance measures

We evaluated the performance of sequences labeled with single classes through calculation of precision, recall (sensitivity), and the $F_1$ measure. The recall for class $c_j$, denoted $s_j$, is defined as the fraction of proteins belonging to class $c_j$ that were predicted correctly. The precision for class $c_j$, denoted $p_j$, is defined as the fraction of proteins predicted to belong to class $c_j$ that were correct predictions. We define the following:

- $TP_j$ – Number of correctly predicted sequences belonging to class $c_j$,
- $FN_j$ – Number of incorrectly predicted sequences belonging to class $c_j$,
- $TN_j$ – Number of correctly predicted sequences not belonging to class $c_j$,
- $FP_j$ – Number of incorrectly predicted sequences not belonging to class $c_j$.

In these terms, we define $s_j$ and $p_j$ as:

$$s_j = \frac{TP_j}{TP_j + FN_j}, \qquad p_j = \frac{TP_j}{TP_j + FP_j}.$$

The $F_1$ measure is a per-class measurement that combines recall (sensitivity) and precision on the specified class in the calculation. $F_1$ for class $c_j$ is defined as:

$$F_{1j} = \frac{2 s_j p_j}{s_j + p_j}.$$

For determining overall classifier performance, we report micro-averaged and macro-averaged $F_1$, denoted $micF_1$ and $macF_1$ respectively. These measures are used widely in evaluation of methods for text classification, particularly in research on unbalanced data [54]. They give a single overall measure of classifier performance, particularly in the midst of unbalanced, multi-class data. With micro-averaged scores, the measures are produced by totaling the appropriate counts with respect to each class, and then calculating the measure. In this way, an equal weight is given to every protein instance, regardless of class. Micro-averaged scores tend to be dominated by the performance of the large classes in the data. In comparison, macro-averaged scores are calculated on a per-class basis first, and then the measure is averaged across all classes to produce the final measure. Here, an equal weight is given to every class, regardless of how often instances in any specific class occur in the data. Macro-averaged scores are dominated by the performance of the classifier on rare classes [55]. Because we are particularly interested in improving predictions on the smaller represented classes, we focus mostly on macro-averaged $F_1$. It provides the most honest measure of performance across all classes without artificially inflating the per-

formance measure due to over-predictive behavior of the classifier on the large classes. However, we report micro-averaged $F_1$ for completeness. It can be easily shown that micro-averaged $F_1$ is equivalent to the total accuracy for the classifier. However, because the majority of our data are unbalanced, we believe that micro-averaged measures, such as $micF_1$, commonly used in a wide array of research on classification, fail to reveal the performance under this situation.

All performance measures are based on a standard 10-fold cross validation, where the dataset is broken up into 10 randomly selected disjoint partitions. The model is trained with nine parts and tested with the remaining part. This is repeated over all 10 possible combinations.

### 6.3. Empirical results

We present the results of different experiments conducted using our datasets that show how unlabeled data can be used to improve predictive performance over a variety of scenarios encountered in protein classification. We will state the value of $n$ for the $n$-gram length chosen for each test. It is important to choose $n$ in such a way that the classifier is able to generalize beyond the training data. If you have access to large amounts of labeled data, then larger values of $n$ (typically $n \geqslant 5$) will yield better results on new data yet to be seen. If $n$ is too large, then the feature space will consist of unique $n$-grams that were observed in the training data, thereby overfitting the training data. If $n$ is too small, then the feature space will be too restrictive for learning discriminative features in the data [4]. Because we reduce the similarity to a maximum threshold of 60% sequence identity in our data, and because we randomly select small subsets of the dataset, the actual level of similarity in most tests will be much less than 60%, and thus smaller values of $n$ (usually ranging from 4 to 6) will be used. The optimal value of $n$ is chosen based on 10-fold cross validation using the ngLOC method without unlabeled data. All subsequence tests with unlabeled data are based on a 10-fold cross validation unless otherwise stated.

#### 6.3.1. Basic-EM results

Our first test was designed to confirm the usefulness of the basic EM approach. We tested the $D_L$-2A dataset over different sizes of labeled data, using a fixed-size set of 25,000 unlabeled data sequences selected at random from $D_U$. An $n$-gram size of 6 was chosen for this test. Figure 5 shows a graph of the macro-averaged $F_1$ measure with and without unlabeled data being used. Table 2 reports both the macro-averaged $F_1$ ($macF_1$) and micro-averaged $F_1$ ($micF_1$) measures generated on different fractions of the $D_L$-2A dataset that result with and without 25,000 unlabeled data sequences.

The EM algorithm used with the unlabeled data results in significantly improved $macF_1$ and $micF_1$ measures for all fractions of the labeled data used. As expected, the improvement is much more dramatic when only a small fraction of the unlabeled data are used.

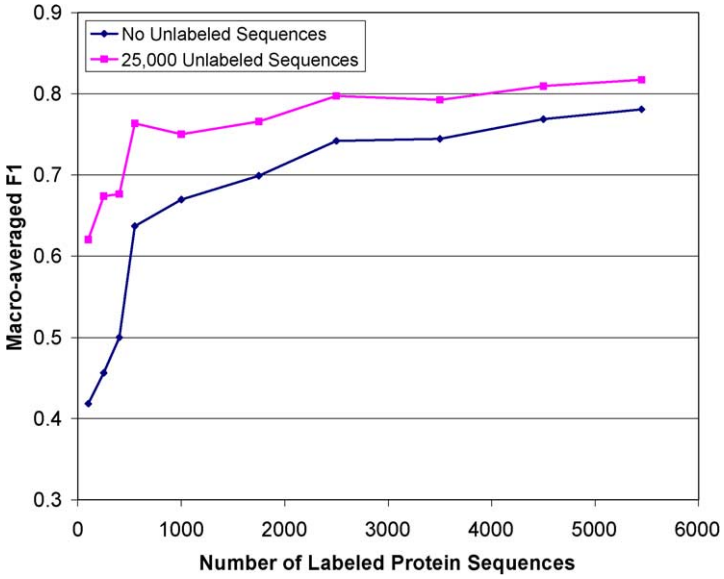


Fig. 5. Basic EM algorithm on 2-class dataset.

Table 2
Performance improvement on 2-class data. This chart displays how different fractions of the unlabeled data show remarkable improvement as a result of applying the basic EM algorithm with 25,000 unlabeled sequences

| # Labeled seqs | No UL data | | 25,000 UL seqs | |
|---|---|---|---|---|
| | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ |
| 100 | 0.418 | 0.449 | 0.620 | 0.622 |
| 250 | 0.457 | 0.474 | 0.674 | 0.677 |
| 400 | 0.500 | 0.517 | 0.676 | 0.676 |
| 550 | 0.637 | 0.649 | 0.764 | 0.764 |
| 1,000 | 0.670 | 0.680 | 0.750 | 0.752 |
| 1,750 | 0.699 | 0.717 | 0.766 | 0.772 |
| 2,500 | 0.742 | 0.748 | 0.797 | 0.799 |
| 3,500 | 0.744 | 0.752 | 0.792 | 0.795 |
| 4,500 | 0.769 | 0.775 | 0.810 | 0.812 |
| 5,450 | 0.781 | 0.787 | 0.817 | 0.820 |

When there are only 100 labeled sequences, $macF_1$ improves from 0.418 to 0.62 and $micF_1$ improves from 0.449 to 0.622, resulting in a remarkable 48% and 38% improvement in the measures, respectively. However, even when all 5,450 labeled sequences are used, we still observe an improvement in the $macF_1$ measure, increasing by 4.6% from 0.781 to 0.817. Additionally, $micF_1$ improved 4.2% from 0.787 to 0.82. While these performance improvements are small, it shows that this approach is useful even when the amount of labeled data is relatively large. This behavior suggests that as the number of labeled sequences increase in the model, there are fewer discriminatory $n$-grams to be discovered in the data, and may justify how the improvement in predictive performance decreases proportionally to the amount of unlabeled data used.

#### 6.3.2. EM-$\lambda$ results

It is possible to overwhelm the classifier with too much unlabeled data, resulting in skewed parameter estimates for the model which becomes detrimental to predictive performance [5]. We chose a random selection of 250 sequences from the $D_L$-2A, and using an $n$-gram length of 4, we applied the basic EM algorithm while varying the fraction of unlabeled data. Table 3 reports the results, and shows how both $macF_1$ and $micF_1$ begin to deteriorate when the unlabeled data becomes increasingly large. For this particular test, we observed the best performance when applying 25,000 unlabeled sequences. At this point, we observed a $macF_1$ measure of 0.674, a 47% improvement from 0.457, that was achieved without using any unlabeled data. Using more unlabeled data caused the performance to decrease, dropping by more than 4% from the best observed measure for this test. This shows that it is possible to overwhelm the model parameter estimates with too many expected predictions from the unlabeled data. It is for this reason that the EM-$\lambda$ modification was made, to reduce the effect that the unlabeled data will have on determining the parameter estimates.

Table 3
Effect of unlabeled data on classification of 250 sequences from the $D_L$-2A dataset. The numbers in bold indicate the best results observed for this test

| # Unlabeled | $macF_1$ | $micF_1$ |
|---|---|---|
| 0 | 0.457 | 0.474 |
| 2,000 | 0.602 | 0.602 |
| 4,000 | 0.633 | 0.633 |
| 6,000 | 0.637 | 0.637 |
| 8,000 | 0.645 | 0.645 |
| 12,000 | 0.649 | 0.649 |
| 16,000 | 0.653 | 0.653 |
| 20,000 | 0.669 | 0.668 |
| **25,000** | **0.674** | **0.677** |
| 37,500 | 0.650 | 0.653 |
| 50,000 | 0.644 | 0.649 |

For our next test, we used the $D_L$-3 dataset, which consisted of the most rare classes in the original dataset obtained from the ngLOC method [4]. These data consisted of 584 sequences total, of which only 80 are classified as being targeted to LYS, representing only 13.7% of the data. The largest class in this dataset consists of those sequences targeted to GOL, which represents over 64% of the data. The remainder of the data are targeted to POX. This dataset exhibits numerous challenging characteristics common in real-world data classification problems: the amount of labeled data is extraordinarily small compared to other localization classes, and the data are highly unbalanced across all classes. A primary aim in this research was to improve predictive performance on rare classes that are also unbalanced, and these tests are designed to address these challenges.

We observed in our previous test how too much unlabeled data can have an adverse effect in situations where the amount of labeled data is a small fraction of the unlabeled data. We tested the EM-$\lambda$ algorithm [5] with the entire $D_L$-3 dataset over different amounts of unlabeled data and different values of $\lambda$. We used an $n$-gram length of 4 for this test. Table 4 shows the results of this test. The baseline results for this test with no unlabeled data are $macF_1 = 0.561$ and $micF_1 = 0.728$.

Table 4

The performance results for the $D_L$-3 dataset across differing values of the $\lambda$ parameter in the EM-$\lambda$ algorithm and differing amounts of unlabeled data. The numbers in bold indicate the best test results for a given quantity of unlabeled data (Baseline results with no unlabeled data are $macF_1 = 0.561$ and $micF_1 = 0.728$.)

| $\lambda$ | 2,000 UL sequences | | 4,000 UL sequences | | 8,000 UL sequences | | 25,000 UL sequences | | 50,000 UL sequences | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ |
| 0.01 | 0.581 | 0.735 | 0.592 | 0.738 | 0.598 | 0.741 | 0.635 | **0.745** | **0.677** | **0.752** |
| 0.025 | 0.598 | 0.741 | 0.592 | 0.736 | 0.608 | 0.736 | **0.674** | 0.738 | 0.657 | 0.700 |
| 0.05 | 0.591 | 0.736 | 0.606 | 0.735 | 0.647 | 0.740 | 0.659 | 0.700 | 0.623 | 0.654 |
| 0.075 | 0.602 | 0.735 | 0.633 | **0.743** | **0.672** | **0.747** | 0.627 | 0.661 | 0.613 | 0.640 |
| 0.1 | 0.606 | 0.735 | 0.631 | 0.733 | 0.669 | 0.729 | 0.627 | 0.658 | 0.610 | 0.637 |
| 0.25 | 0.653 | **0.741** | **0.674** | 0.729 | 0.637 | 0.673 | 0.618 | 0.647 | 0.622 | 0.652 |
| 0.5 | **0.673** | 0.733 | 0.649 | 0.692 | 0.625 | 0.656 | 0.622 | 0.656 | 0.633 | 0.668 |
| 0.75 | 0.663 | 0.719 | 0.643 | 0.683 | 0.629 | 0.659 | 0.626 | 0.663 | 0.643 | 0.682 |
| 1 | 0.656 | 0.711 | 0.643 | 0.682 | 0.637 | 0.673 | 0.631 | 0.670 | 0.656 | 0.697 |

The most important observation to make from this test is with regard to the effect that the $\lambda$ parameter has over different amounts of unlabeled data. Through selection of appropriate values of $\lambda$, the unlabeled data can be controlled and prevented from dominating the parameter estimates. The best results were obtained when using 50,000 unlabeled sequences, and choosing $\lambda = 0.01$. This has an effect of reducing the ratio of unlabeled-to-labeled sequences to be roughly one unlabeled instance to every one labeled instance. While this may seem ideal, this should not be interpreted as a rule. (Refer to the discussion section for more details.)

We again notice that the $macF_1$ measure improved in every single test executed, indicating that the performance on the smallest classes in this dataset improved. Though there were numerous tests where $micF_1$ decreased below the 0.728 measure established without any unlabeled data, this did not happen on the best performing tests. (Regardless, we note that this is not necessarily a bad thing, because even though predicting every instance as GOL would yield a reasonable $micF_1$ measure of 0.647, this same classification would result with a $macF_1$ of 0.262, stressing the importance of observing $macF_1$ on unbalanced datasets.) We are more interested in improving the sensitivity on the smaller classes, ideally without a decrease in total accuracy, which was indeed accomplished. The best results yielded a more than 20% improvement in $macF_1$, increasing from 0.561 to 0.677, while showing a more than 3% improvement in $micF_1$, increasing from 0.728 to 0.752. This showed that using more unlabeled data was able to significantly improve predictive performance of most rare classes in the data, as long as we were able to control the dominating effect that the unlabeled data had through the appropriate selection of $\lambda$.

### *6.3.3. Multi-EM results*

For our next test, we used the $D_L$-2B class dataset to evaluate the performance of the multi-component mixture model (Fig. 3). We chose this dataset for this test, because we know that both the nucleus (NUC) and mitochondria (MIT) organelles in the cell are comprised of distinct suborganelles. As discussed, we would expect a multi-component mixture model to be able to model this better. For this test, we use $n$-gram values of both 4 and 5 for comparative purposes, and evaluate different numbers of components selected per class. We use a randomly selected set of 50,000 unlabeled data sequences, Table 5 shows the results of this test.

All tests conducted yielded significantly better results than not using any unlabeled data, particularly with the $macF_1$ measure, indicating that the smallest classes in the data experienced remarkable improvement in sensitivity. The best performing tests were those that used more than one component per class in almost every case. The only exception was the $macF_1$ measure observed on the set of 1,250 labeled sequences using the 5-gram model, where the single component model surprisingly resulted in a better $macF_1$ measure (0.725) than the multi-component model. We ran further tests using larger amounts of labeled data, and we continued to observe a better $macF_1$ measure on the single component model. The 4-gram model did not have this problem on larger size labeled datasets tests. In all tests, the best $micF_1$ was always observed for the multi-component model.

These tests showed us that the multi-component model does indeed capture the underlying distribution of discriminating $n$-grams at the subcomponent

Table 5

Results for the Multi-EM algorithm. This table presents the results from using varying amounts of multiple components per class in the mixture model (denoted *numComp*) on the $D_L$-2B dataset, using both a 4-gram and 5-gram feature space. We tested across different number of labeled sequences, using 50,000 unlabeled sequences in the test. The numbers in bold indicate the best performing results for a given number of labeled sequences. The results in italics represent the baseline results without unlabeled data

| # Labeled seqs | *numComp* | 250 | | 750 | | 1,250 | |
|---|---|---|---|---|---|---|---|
| | | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ |
| 4-gram | | *0.4871* | *0.7406* | *0.5959* | *0.7940* | *0.5679* | *0.8065* |
| | 1 | 0.6967 | 0.7669 | 0.7309 | 0.7966 | 0.7459 | 0.8195 |
| | 2 | 0.6972 | 0.7707 | **0.7417** | 0.8057 | 0.7463 | 0.8211 |
| | 3 | **0.7007** | **0.7744** | **0.7411** | **0.8070** | **0.7475** | **0.8236** |
| | 4 | 0.6972 | 0.7707 | 0.7402 | **0.8070** | 0.7462 | **0.8236** |
| 5-gram | | *0.5386* | *0.7594* | *0.5639* | *0.7849* | *0.5326* | *0.8033* |
| | 1 | 0.6625 | 0.7256 | 0.7011 | 0.7640 | **0.7250** | 0.8033 |
| | 2 | 0.6543 | 0.7368 | 0.7192 | 0.7979 | 0.7094 | 0.8154 |
| | 3 | 0.6568 | 0.7519 | **0.7210** | **0.8162** | 0.6945 | **0.8244** |
| | 4 | **0.6693** | **0.7744** | 0.6947 | 0.8083 | 0.6689 | 0.8211 |

level better than using a single component per class. This was much more evident with the 4-gram model, likely because the 4-gram model does a better job at being able to generalize beyond the observed training data. This is confirmed by comparing the performance measures without any unlabeled data between the 4-gram and 5-gram. Introducing multiple components per class into the mixture model introduces an extra risk of reducing generalization on new data not yet observed. For this reason, the smallest possible $n$-gram that can effectively classify the data should be used.

### 6.3.4. EM-CS results

For our next set of tests, we evaluated the EM-CS algorithm, which iteratively constructs a subset of unlabeled sequences to be used for estimating model parameters by selecting sequences at each iteration that meet or exceed a predetermined confidence score threshold. We chose the $D_L$-3 dataset for this test because it was designed to contain the rarest classes observed across all possible classes that could be observed, and thus it is quite reasonable to assume that the majority of the data in the unlabeled dataset are not associated with any of these three classes.

For this test, we chose a 4-gram model. We define *CSthresh* to be the minimum allowed confidence score (*CS*) that a sequence must have to be considered in the model. For test purposes, we first generated predictions without using any unlabeled data on all the data in $D_L$-3. We observed a *CS* range between 33.5 and 47.8 on all the data. Table 6 reports the percentile of the data falling over different selected *CS* thresholds in this test.

Table 6

Percentile of data in DL-3 falling in different ranges of *CS* based on cross validation. The predictions were generated using a 4-gram model without using any unlabeled data. The range of the *CS* for all predictions falls between 33.5 and 47.8. We report the percentile to indicate the fraction of data in the $D_L$-3 data that is falling below the specified *CS* value. For example, 71% of all predictions in the $D_L$-3 dataset score less than 37

| *CS* | % Correct | Percentile |
|---|---|---|
| 40 | 100 | 94 |
| 38.25 | 97 | 87 |
| 37 | 88 | 71 |
| 36 | 85 | 50 |
| 35 | 78 | 21 |

Next, we evaluated the EM-CS algorithm at these different *CS* thresholds using a dataset of 60,000 unlabeled sequences. The performance results from the test are shown in Table 7. We report the number of sequences actually used in the model for each value of *CSthresh* used. We also report the results without using any unlabeled data, and the results of using all the unlabeled data for comparison purposes.

As expected, a large number of sequences are able to be eliminated from the unlabeled dataset. Our lowest *CSthresh* value of 35 had eliminated almost 89% of the data in the unlabeled dataset. This confirmed that an overwhelming majority of this dataset is not likely to belong to these three classes. More importantly, we also observed an improvement in the $macF_1$ and $micF_1$ measurements by eliminating these sequences.

Table 7

The results for the EM-CS algorithm are displayed. The value of *CSthresh* was selected at different levels ranging between 35 and 40. The number of unlabeled sequences that were actually selected is reported. The numbers in bold indicate the best results. The numbers in italics indicate the baseline results for a model generated without unlabeled data

| *CSthresh* | Unlabeled sequences | $macF_1$ | $micF_1$ |
|---|---|---|---|
| | 0 | *0.561* | *0.728* |
| 0 | 60,000 | 0.650 | 0.690 |
| 35 | 6,814 | 0.662 | 0.721 |
| 36 | 3,810 | **0.674** | 0.735 |
| 37 | 2,388 | **0.676** | 0.741 |
| 38.25 | 1,474 | **0.672** | **0.752** |
| 40 | 726 | 0.621 | 0.740 |

The best results (indicated in bold in the table) occurred with a *CSthresh* value of 38.25, which resulted in 97.5% of the unlabeled data being eliminated. In this test, the $macF_1$ measure improved to 0.672 from 0.561, and $micF_1$ improved to 0.752 from 0.728. The scores improved over the results obtained from using the entire unlabeled dataset in all test cases, with a notable exception when *CSthresh* $=$ 40, demonstrating how performance can degrade if there are too few sequences selected due to a more restrictive setting of *CSthresh*.

### 6.3.5. Comparison against transductive SVM

For our final set of tests, we evaluated our method against the transductive SVM (TSVM) method discussed above [7,33]. We conducted a test on the $D_L$-2A dataset to ensure that the TSVM would yield reasonable results on a binary, balanced dataset. Our second test was on the $D_L$-3 dataset, which represents the most challenging problems for protein sequence classification.

The TSVM performed well on the basic $D_L$-2A dataset, but did not perform well on the $T_L$-3 dataset. The results for the $D_L$-2A data are shown in Table 8 and are presented first. The SVM method performed remarkably well, resulting in a $macF_1$ measure of 0.789. Adding unlabeled data for the TSVM method only marginally improved the results to almost 0.81 by including 12,000 unlabeled sequences. We noticed the predictive performance start to drop with any additional data beyond this. Moreover, we could not consider using more than 15,000 unlabeled sequences because the TSVM would not converge on a solution. The EM-CS algorithm showed a more remarkable improvement from no unlabeled data, increasing the $macF_1$ measure from 0.624 to 0.789 with 60,000 unlabeled sequences, resulting in a 26.4% improvement. However,

Table 8

The results of comparing TSVM against the EM-CS algorithm on the $D_L$-2A balanced, binary class dataset. The results in bold are the best results from each method. NA $=$ Not available

| # UL sequences | TSVM | | EM-CS | |
|---|---|---|---|---|
| | $macF_1$ | $micF_1$ | $macF_1$ | $micF_1$ |
| 0 | *0.7894* | *0.7894* | *0.6244* | *0.6518* |
| 2,000 | 0.8004 | 0.8004 | 0.7479 | 0.7555 |
| 4,000 | 0.8022 | 0.8022 | 0.7614 | 0.7685 |
| 6,000 | 0.8059 | 0.8058 | 0.7538 | 0.7611 |
| 8,000 | **0.8095** | **0.8095** | 0.7552 | 0.7629 |
| 12,000 | **0.8095** | **0.8095** | 0.7681 | 0.7759 |
| 15,000 | 0.8059 | 0.8059 | 0.7723 | 0.7796 |
| 60,000 | NA | NA | **0.7889** | **0.7826** |

Table 9

Comparing the actual running time for a 10-fold cross validation on the $D_L$-2A dataset over different amounts of unlabeled data between both methods. The results reported are in actual seconds of CPU time. All tests were conducted on a 3.0 GHz Intel x64

| # UL sequences | TSVM (s) | EM-CS (s) |
|---|---|---|
| 0 | 6 | 19 |
| 2,000 | 555 | 468 |
| 4,000 | 2,016 | 1,147 |
| 6,000 | 4,768 | 1,609 |
| 8,000 | 9,908 | 3,090 |
| 12,000 | 14,653 | 5,069 |
| 15,000 | 122,813 | 9,326 |

we point out that the standard inductive SVM method performed the same as our method with unlabeled data, and adding unlabeled data marginally improved the results for the TSVM method. On the best results, the TSVM with 8,000 unlabeled sequences outperformed the EM-CS algorithm by 2%.

There is a substantial cost in the computational time required for the TSVM compared to our method. See Table 9 for the results. We note that the computational time required for our method to complete is linear with respect to the size of the unlabeled dataset. SVM methods are known to have a running time between $O(n^2)$ and $O(n^3)$ with respect to the number of examples $n$ in the labeled and unlabeled data, dependent on the value of the $C$ and $C^*$ parameters chosen [56]. Our empirical results shown in Table 9 clearly confirm these running times. This represents a significant benefit of our method over the TSVM. See the discussion section for more information pertaining to these results.

The results from our EM-based methods were clearly superior compared to the results from the TSVM on the DL-3 data. We report only the results from the

Table 10
Results of EM-$\lambda$ against the transductive SVM on the $D_L$-3 dataset. Results in bold indicate the best results observed for each method. NA = not available

| # UL sequences | TSVM | | EM-$\lambda$ | |
|---|---|---|---|---|
| | *macF*$_1$ | *micF*$_1$ | *macF*$_1$ | *micF*$_1$ |
| 0 | *0.499* | *0.626* | *0.561* | *0.728* |
| 2,000 | 0.641 | 0.653 | 0.602 | 0.735 |
| 4,000 | 0.645 | 0.661 | 0.633 | 0.743 |
| 6,000 | **0.659** | **0.667** | 0.645 | 0.743 |
| 8,000 | 0.654 | 0.661 | 0.672 | 0.747 |
| 50,000 | NA | NA | **0.677** | **0.752** |

EM-$\lambda$ method. The results are displayed in Table 10. The best results from the TSVM method were obtained from using 6,000 unlabeled sequences, where the *macF*$_1$ measure increased from 0.5 without any unlabeled sequences to 0.66 by using unlabeled data, yielding a 32% increase in performance. Additionally, the *micF*$_1$ (overall accuracy) measure increased from 0.626 to 0.677, an 8.1% increase on the same test. As noted in the previous test, using more unlabeled data resulted in performance degradation. The TSVM failed to converge when using more than 10,000 unlabeled sequences. The best results from the EM-$\lambda$ method were obtained from using 50,000 unlabeled sequences, where the *macF*$_1$ measure increased from 0.561 to 0.677, yielding a 21% performance improvement based on this measure. The *micF*$_1$ measure increased from 0.728 to 0.752, yielding a 3.3% increase in the measure. We note that our methods outperformed the TSVM in all cases, with the *macF*$_1$ measure showing a marginal 2.7% improvement over TSVM. We were pleased to see a significant 12.7% improvement in the overall accuracy from our method compared to the TSVM. The results of these two tests suggest that the TSVM performs well on simple binary, balanced classification problem; however the EM-based approach is more useful for multi-class problems where the data are unbalanced and availability of data are limited. See the discussion for a more detailed exposition on these methods.

## 7. Discussion

Our previous work on the ngLOC method [4] has shown that when there is plenty of labeled data available, and there is a relatively high similarity threshold that is allowed in the training data, classification performance on new data can be quite impressive. However, in our previous work, as is the case with many other classifiers in this domain, prediction tends to suffer on the classes in the data with significantly low representation. Furthermore, there are significantly more biological concepts that proteins can be classified with that have significantly less data available than subcellular localization. Traditional supervised learning methods require a sufficient amount of labeled training data for inference of a model that has good generalization performance. In this study, we showed remarkable improvement in performance over supervised methods (i.e., naïve Bayes and SVM) when learning from small, multi-class, unbalanced datasets. Moreover, we gained the most substantial improvement in the least represented classes. These are challenges which are likely to be characteristic of a wide range of other protein annotations beyond subcellular localization. This suggests the value of this work for inducing models of protein characteristics that have very little labeled data available, perhaps allowing the inference of classification models on functional aspects of proteins that have yet to be considered.

### 7.1. Comparison against text classification methods

We applied the naïve Bayes classifier using an $n$-gram model for classification, where each possible $n$-gram is the equivalent of a word in a document used in text categorization. The largest distinguishing difference between our work and the feature space commonly found in text classification is the size of the feature space, denoted $|\mathcal{X}|$. Typical feature spaces in text categorization run on the order of $10^3$–$10^5$ distinct features. In contrast, the feature space in protein sequence classification is exponential with respect to the value of $n$ chosen, where the feature space is $O(20^n)$. While it is true that the observed feature space becomes sparse for values of $n > 4$, it is still enormous. Even at $n = 4$, there are 160,000 possible features, with almost all of them observed in the training data. At $n = 5$, though there are 3.2 million theoretically possible features, there were roughly 2.2 million features observed. Through careful, efficient memory management, and accounting for only those $n$-grams that occur in the training data, larger values of $n$ are still very feasible to use. The primary restriction with using large values of $n$ is the fact that it significantly reduces generalization beyond the training data. In general, if there is access to a large set of training data, and/or all data in the domain of interest have a very high level of sequence similarity, then large values of $n$ are needed to repre-

sent discriminating $n$-grams that can effectively distinguish between the classes in the dataset. We use cross-validation to select the most ideal size of $n$ for each dataset used.

The $n$-gram model represents a powerful, yet simple feature space for protein sequences. This model is able to capture sequence homology, while allowing for differences due to insertion, deletion, and/or mutation. It effectively shrinks the protein sequence space, thereby allowing a higher degree of redundancy between proteins that could not be had by considering the entire protein sequence. Moreover, different sized $n$-grams of significant length will map to structural features of the protein differently. Secondary structure elements are vital for attaining a proper fold of a protein, and consequently are vital for its function. Hence, these secondary structures are distinctly conserved across proteins with different functions. Basic amino acid or pairwise amino acid composition (i.e., a 1-gram or 2-gram model) represent feature spaces frequently used in this field, most likely due to the small dimensionality that is required by many classification methods. Though it has been applied with varying degrees of success, this limited feature space will fail to capture distinctive structural components of the protein. As demonstrated, a naïve Bayes classification method such as ours does not have this restriction. Fixed-length $n$-gram models were considered for this study, though numerous extensions to the basic $n$-gram model (e.g., single-gapped $n$-grams, variable-gapped $n$-grams) could be considered.

A significant difference with our work compared to other semi-supervised work in text classification is in regard to the domain in which the unlabeled data are taken from. In most semi-supervised work in text classification, it is assumed that the unlabeled data comes from the same source as the labeled data, and that the only difference is that it has not been categorized. Probabilistically speaking, it is assumed that $D_U$ is drawn from the same distribution as $D_L$, formally stated as $\forall c_j \in \mathcal{C}, P(c_j|D_L) \approx P(c_j|D_U)$. However, many methods do not consider that $D_U$ may be drawn from a different distribution. Even worse, $D_U$ may contain documents belonging to classes entirely outside of the target classes. For example, if documents belonging to different sports from various athletic magazines are being categorized, documents pertaining to financial or business subject matters are not incorporated in $D_U$. This is a very reasonable assumption for most problems in text categorization; however, it is not reasonable for the majority of semi-supervised protein classification tasks. While it is true that, biologically speaking, all proteins are synthesized through the same process of DNA replication, transcription, and translation, the mere fact that a protein has been successfully synthesized from a common source says absolutely nothing about its functional role in a species, or where in the cell it needs to localize to perform its function. It is for this reason that the EM-CS enhancement to the basic EM algorithm was added, to increase the likelihood of incorporating sequences that may at least have some association with one of the classes of interest. (The EM-CS algorithm is discussed below.)

### 7.2. Complexity of naïve Bayes and EM

The space requirements can be rather extensive for larger $n$-grams, and is upper-bounded by $O(|\mathcal{C}|20^n)$. The Multi-EM extension adds an additional multiplicative factor of the number of components selected per class. However, as discussed, for values of $n \geqslant 5$, the observed feature space becomes very sparse, with only a small fraction of all theoretically possible $n$-grams observed. Therefore it is essential that a one-to-one mapping of $n$-grams to distinct indices is maintained, in order to allow the exploration of large $n$-grams. However, the run time is linear with respect to the size of the labeled and unlabeled data – a very nice benefit of using the naïve Bayes classifier. If there are $m_L$ sequences in $D_L$, and $m_U$ sequences in $D_U$, and the average number of EM-iterations before convergence is $p$, and the average length of a sequence is $k$ amino acids, then the running time for learning is bounded by $O(kp(m_L + m_U))$.

### 7.3. Considerations of EM enhancements

In the EM-$\lambda$ algorithm, the $\lambda$ parameter serves as a weight applied to unlabeled data, in order to control the effect that unlabeled data has in determining parameter estimates. It explicitly violates the assumption that each sequence is generated by one component in the model, by assuming that a sequence can be generated by a fraction of a component. While this assumption does not seem sound, it is a very reasonable in situations where the amount of unlabeled data would render the labeled data ineffective in determining the final parameter estimates. We clearly showed how too much unlabeled data causes the predictive performance to deteriorate (Table 3), and we are quite certain that even more unlabeled data beyond what is shown would not

improve the situation. As a final note regarding this algorithm, using EM-$\lambda$ in conjunction with EM-CS will result in significantly less protein sequences from the unlabeled data selected, and therefore a larger $\lambda$ parameter will likely be required. Our own observations confirmed this behavior (data not shown).

We point out that in several of our tests of the EM-$\lambda$ algorithm, we often observed the best results when selecting a value for $\lambda$ such that the unlabeled data was equally weighted with respect to the labeled data. However, this was not always observed, and should not be used as anything more than a good starting point, and then exploring alternative settings from there. Cross-validation can be used to select the most ideal value of $\lambda$ for classification of new data.

In general, the problem of controlling the undesired influence of unlabeled data on parameter estimates can be addressed through either the EM-$\lambda$ or EM-CS algorithms. The EM-CS will offer substantial running time improvement that EM-$\lambda$ will not have. Moreover, EM-CS does not require relaxing the assumption that each component generates one sequence, unlike EM-$\lambda$ does, because the unlabeled data are controlled by removing poor examples of the data, instead of application of the weight parameter to reduce the effect of all unlabeled data. However, if it is determined that a large fraction of the unlabeled data contain instances belonging to the target classes, then EM-$\lambda$ is the algorithm of choice, otherwise, EM-CS will likely yield superior results, provided an appropriate value of *CSthresh* is chosen.

The EM-CS extension offers the most benefit when there is high certainty that the majority of the data in the unlabeled dataset are likely to belong to other classes not in $\mathcal{C}$. We successfully showed predictive performance improvements through the test on the $D_L$-3 dataset, which contained unbalanced data consisting of proteins localized to three distinct organelles that have a relatively small number of sequences targeted to them over all protein sequences available. Without the EM-CS extension, sequences that will never localize into these three organelles will be incorporated into the model, likely skewing parameter estimates because there are significant quantities of sequences not belonging to any target class.

In the EM-CS algorithm, we witnessed the importance of incorporating the sequences meeting the specified CS threshold in proportion to the current estimates for the class prior distribution (data not shown). Specifically, this requirement was most useful in situations where data are highly unbalanced across classes. Without this restriction, we observed tests where the class prior distribution would quickly become more unbalanced than it was before beginning EM iterations, resulting with the prior probability becoming increasingly weighted toward the classes that were already the most populated.

We showed how the performance of EM-CS was dependent on the value of *CSthresh* chosen. By observing the results in both Table 6 and Table 7, there is a clear tradeoff between choosing an appropriate value for *CSthresh* that maximizes selection of the most probable sequences that belong to one of the target classes, while allowing selection of enough unlabeled data to use in improving parameter estimates. A *CSthresh* value that is too high will allow many potentially useful sequences to be prevented from being used in the EM-CS algorithm, while a *CSthresh* value that is too low will incorporate sequences that are likely to belong to other classes outside of the set of target classes being modeled. The best selection of *CSthresh* is chosen through cross validation based on observation of scores without unlabeled data first, and then choosing a few scores around the score falling at the 50 percentile mark.

A small improvement could be made to the EM-CS algorithm to improve the running time by limiting the number of iterations that the algorithm is allowed to add new sequences to the selected subset. We rarely observed any significant additional sequences added after the first few iterations of the EM algorithm. By eliminating the need to calculate the expected class assignments on the entire unlabeled data on future iterations of the E-step (and thereby eliminating those sequences from the M-step), the computational time required could be significantly improved without degradation in predictive performance.

In the multiple-component mixture model, there are varying techniques to consider for selecting the best number of components to model each class, including cross-validation, Akaike information criterion (AIC) and Bayesian information criterion (BIC) [3,13]. We used cross validation to evaluate which number of components were performing the best. However, the results we presented are based on a restricted implementation which uses a fixed number of components for every class. While we believe that there may be a benefit to selecting a specific number of components for each class, this was not tested.

### *7.4. Comparison against transductive SVM*

A significant problem with SVM-based methods is the computational resources required in training the

methods, particularly when considering non-linear kernels in the midst of training with high-dimensional, large-scale datasets. Our results showed how the computational time required to train transductive SVMs increased dramatically with respect to the amount of unlabeled data incorporated (see Table 9). Moreover, the method would often fail to converge if too much unlabeled data were used. Sometimes clustering techniques are applied and selective instances are used to represent each class, in order to reduce the size of the data used.

For high-dimensional data in particular, SVM methods become intractable for all but the smallest datasets. Solutions involve implementation of feature selection methods to reduce the feature space, clustering techniques to reduce the high-dimensionality of the data, or methods to reduce the size of the data. The computational resources required by our methods for training and testing are linear with respect to the dimensionality and amount of data used, which remains quite tractable for very large, high dimensional data.

Many existing SVM-based methods for protein sequence classification have used variants of standard amino acid composition and occurrence of amino acid pairs for feature representation [57,58]. These are equivalent to a 1-gram and a 2-gram model in our work. As noted previously, this feature space is very limited, particularly for work that requires discrimination across multiple classes, as it will not capture the discriminatory $n$-grams required to distinguish between classes. Despite this, we tested both of these representations for all SVM and TSVM tests conducted. We tried to test 3-gram and higher feature spaces that were used for our methods, but the predictive performance for the SVM methods degraded substantially for the majority of the tests, or the computational time required for the method to converge was unreasonable.

A more significant problem involves the selection of the SVM and TSVM parameters. It is difficult to know whether the parameters you have selected are optimal. Moreover, tuning of the parameters usually involves a manual process of repeated tests until you achieve superior results, which can be an exhaustive task. This is a well known challenge with SVMs [59], and represents an ongoing area of research in the machine learning community. Despite this challenge, when acceptable parameters are found, the results can be very good.

Quite often, the data are projected into a higher or different dimensional space through the kernel trick. It is possible that an ideal kernel may be found, and excellent SVM parameters might be determined for a classification problem, but impossible to understand how the translated feature space is discriminating between classes. With our methods, the feature space can be explored to extract features that were significant for classification, or anomalies in the data that were prohibitive to correct classification. These are significant requirements in the biomedical research community.

A probabilistic measure ought to be considered a crucial output of any predictive model, and this is a difficult feat due to their non-probabilistic output. Solutions to this problem for SVM methods usually involve creating another model to simulate a probability distribution from the output of the SVM [60]. Alternative techniques sometimes involve implementation of a bootstrap method that create multiple SVMs using random draws from the training data, and estimating a probability distribution based on the output of these random SVMs. In contrast, probabilistic methods such as ngLOC [4] and the work presented in this study allow the output of a probabilistic measure. This is particularly important in biomedical research, where it is usually much more important to attain high precision over maximum overall sensitivity (i.e., recall) for a given classification method.

Though we noted an increase in performance for transductive SVMs compared to inductive SVMs for most of our tests, we noted a decrease in predictive performance for transductive SVMs compared to our semi-supervised method for most tests conducted except for the tests performed on the $D_L$-2A binary class, balanced dataset. Multiple kernels were tested for these tasks over a wide range of parameters, and the linear kernel always resulted in the best performance.

Despite these challenges with inductive and transductive SVMs, they have been shown to be successful on an extremely wide range of problems over many domains. We encourage the reader to consider TSVMs, particularly if an SVM approach has already been shown to work well. Our results indicated that if the inductive SVM performs well, then there is a high likelihood that incorporating a limited amount of unlabeled data with the TSVM may improve the model. Furthermore, methods are continuing to emerge that directly address many of these limitations mentioned. We were encouraged by the results of the basic $D_L$-2A data and believe that TSVMs may be useful for protein sequence classification under certain scenarios. Moreover, observing the plethora of SVM-based methods for sequence classification will yield an equal number of choices for feature representation that researchers have adopted, and therefore, we believe that improved feature selection might have yielded improved performance for TSVMs.

## 8. Conclusion

Methods in protein classification continue to be researched in earnest, primarily focusing on improving classification of proteins from the sequence alone. Existing research on genomic and proteomic methods have yielded a large amount of protein data, of which only a fraction has been experimentally determined. This domain provides an excellent platform for research into semi-supervised learning methods, which have proven useful in domains where there is a large amount of data, and yet only a fraction of it has been labeled.

Many tasks in protein classification deal with challenges that arise due to limited data being available for learning, and unbalanced classes in the data. We presented a collection of EM-based algorithms designed to address these limitations and challenges through incorporation of unlabeled data. Using the $n$-gram model for the feature space, along with the naïve Bayes classifier, we showed how each of the algorithms can be used to improve classification accuracy over models inferred without unlabeled data. We focused extensively on improving prediction performance for the classes with the smallest representation in the training data, and showed how each algorithm improved the unbalanced predictions, sometimes remarkably well. The improvements in predictive performance were most substantial on the problems where the least amount of labeled data was available.

Limitations of this work were discussed, most of which resulted from the fact that our unlabeled dataset contained instances that belonged to classes that were not in the set of target classes being considered. It was also possible to have too much unlabeled data for adjusting the parameter estimates for the model in situations where there is a minute amount of labeled data. However, by relaxing stated assumptions of the model, notably the one-to-one correspondence between the classes and underlying components in the mixture model, we demonstrated how these problems can be addressed through the EM-$\lambda$ algorithm. We proposed a novel, alternative solution by using the EM-CS algorithm, which has the benefit of allowing the stated assumptions of the model to stay intact, while offering an additional improvement in run-time performance not possible with EM-$\lambda$.

We conducted a thorough comparison of our semi-supervised learning method against the transductive inference method of the transductive support vector machine (TSVM). We noted situations where the TSVM may or may not yield good classification performance compared to our method, with reasons for the observed performance discussed. Substantial improvement was shown by our method compared to the TSVM on the most difficult classification problems. However, we believe that the method is promising for protein sequence classification under certain limitations, and may result in a complementary classifier that can work in conjunction with our method, as well as other experimental methods in this field.

We have demonstrated the added value of semi-supervised learning in this domain, using specific tests designed to replicate common protein classification scenarios characterized by having only limited, highly unbalanced annotated data available. Our results have clearly demonstrated the value of incorporating a large set of unannotated protein sequences, showing how the predictive performance of the resulting classifier in the majority of our tests improved in many of these common scenarios.

## Acknowledgments

This work has been supported by the startup funds to CG from the State University of New York (SUNY) at Albany and the graduate student fellowship provided by the Gen*NY*Sis Center for Excellence in Cancer Genomics, SUNY at Albany.

## References

[1] M. Hamady, T.H. Cheung, K. Resing, K.J. Cios and R. Knight, Key challenges in proteomics and proteoinformatics. Progress in proteins, *IEEE Eng. Med. Biol. Mag.* **24**(3) (2005), 34–40.

[2] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin and G. Sherlock, Gene ontology: tool for the unification of biology. The Gene Ontology Consortium, *Nat. Gen.* **25**(1) (2000), 25–29.

[3] O. Chapelle, B. Scholkopf and A. Zien, *Semi-Supervised Learning*, The MIT Press, Cambridge, MA, 2006.

[4] B.R. King and C. Guda, ngLOC: an $n$-gram-based Bayesian method for estimating the subcellular proteomes of eukaryotes, *Gen. Biol.* **8**(5) (2007), R68.

[5] K. Nigam, A.K. McCallum, S. Thrun and T. Mitchell, Text classification from labeled and unlabeled documents using EM, *Mach. Learn.* **39**(2/3) (2000), 103–134.

[6] K. Nigam, A.K. McCallum and T. Mitchell, Semi-supervised text classification using EM, in: *Semi-Supervised Learning*, O. Chapelle, B. Scholkopf and A. Zien, eds, The MIT Press, Cambridge, MA, 2006, pp. 33–54.

[7] T. Joachims, Transductive inference for text classification using support vector machines, in: *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, Bled, Slovenia, June 27–30, 1999, pp. 200–209.

[8] C.A. Ouzounis, R.M. Coulson, A.J. Enright, V. Kunin and J.B. Pereira-Leal, Classification schemes for protein structure and function, *Nat. Rev.* **4**(7) (2003), 508–519.

[9] S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman, Basic local alignment search tool, *J. Mol. Biol.* **215**(3) (1990), 403–410.

[10] J.D. Thompson, D.G. Higgins and T.J. Gibson, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Res.* **22**(22) (1994), 4673–4680.

[11] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard and C. Chothia, Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods, *J. Mol. Biol.* **284**(4) (1998), 1201–1210.

[12] A. Krogh, M. Brown, I.S. Mian, K. Sjolander and D. Haussler, Hidden Markov models in computational biology. Applications to protein modeling, *J. Mol. Biol.* **235**(5) (1994), 1501–1531.

[13] T.M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997.

[14] A.Y. Ng and M.I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes, in: *Advances in Neural Information Processing Systems*, Vol. 14, The MIT Press, Cambridge, MA, 2002.

[15] T. Jaakkola, M. Diekhans and D. Haussler, A discriminative framework for detecting remote protein homologies, *J. Comput. Biol.* **7**(1/2) (2000), 95–114.

[16] S. Hua and Z. Sun, Support vector machine approach for protein subcellular localization prediction, *Bioinformatics* **17**(8) (2001), 721–728.

[17] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares and D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machines, *Proc. Natl. Acad. Sci.* **97**(1) (2000), 262–267.

[18] B. Rost and C. Sander, Prediction of protein secondary structure at better than 70% accuracy, *J. Mol. Biol.* **232**(2) (1993), 584–599.

[19] J. Selbig, T. Mevissen and T. Lengauer, Decision tree-based formation of consensus protein secondary structure prediction, *Bioinformatics* **15**(12) (1999), 1039–1046.

[20] A. Krogh, B. Larsson, G. von Heijne and E.L. Sonnhammer, Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes, *J. Mol. Biol.* **305**(3) (2001), 567–580.

[21] B.Y. Cheng, J.G. Carbonell and J. Klein-Seetharaman, Protein classification based on text document classification techniques, *Proteins* **58**(4) (2005), 955–970.

[22] E. Eskin and E. Agichtein, Combining text mining and sequence analysis to discover protein functional regions, in: *Proceedings of the 9th Pacific Symposium on Biocomputing*, Hawaii, January 6–10, 2004, pp. 288–299.

[23] A. Hoglund, T. Blum, S. Brady, P. Donnes, J.S. Miguel, M. Rocheford, O. Kohlbacher and H. Shatkay, Significantly improved prediction of subcellular localization by integrating text and protein sequence data, in: *Pacific Symposium on Biocomputing*, 2006, pp. 16–27.

[24] J.C. Whisstock and A.M. Lesk, Prediction of protein function from protein sequence and structure, *Quart. Rev. Biophys.* **36**(3) (2003), 307–340.

[25] M. Szummer and T. Jaakkola, Kernel expansions with unlabeled examples, *Adv. Neur. Inf. Proc. Syst.* **13** (2001), 626–632.

[26] X. Zhu, Z. Ghahramani and J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.

[27] O. Chapelle, J. Weston and B. Scholkopf, Cluster kernels for semi-supervised learning, in: *Advances in Neural Information Processing Systems*, Vol. 15, S. Becker, S. Thrun and K. Obermayer, eds, The MIT Press, Cambridge, MA, 2003, pp. 585–592.

[28] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff and W.S. Noble, Semi-supervised protein classification using cluster kernels, *Bioinformatics* **21**(15) (2005), 3241–3247.

[29] J. Weston, C. Leslie, D. Zhou, A. Elisseeff and W.S. Noble, Semi-supervised protein classification using cluster kernels, in: *Advances in Neural Information Processing Systems (NIPS-2003)*, S. Thrun, S. Saul and B. Scholkopf, eds, The MIT Press, Cambridge, MA, 2004.

[30] H. Shin, K. Tsuda and B. Schlkopf, Protein functional class prediction with a combined graph, in: *Proceedings of the Korean Data Mining Conference*, 2004, pp. 200–219.

[31] Y. Lu, Q. Tian, F. Liu, M. Sanchez and Y. Wang, Interactive semisupervised learning for microarray analysis, *IEEE/ACM Trans. Comput. Biol. Bioinf.* **4**(2) (2007), 190–203.

[32] V. Vapnik, Transductive inference and semi-supervised learning, in: *Semi-Supervised Learning*, O. Chapelle, B. Scholkopf and A. Zien, eds, The MIT Press, Cambridge, MA, 2006, pp. 453–472.

[33] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[34] V.N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[35] M.A. Krogel and T. Scheffer, Multi-relational learning, text mining, and semi-supervised learning for functional genomics, *Mach. Learn.* **57**(1/2) (2004), 61–81.

[36] N. Kasabov and S. Pang, Transductive support vector machines and applications in bioinformatics for promoter recognition, *Neur. Inf. Proc. – Lett. Rev.* **3**(2) (2004), 31–38.

[37] X. Zhu, Semi-supervised learning literature survey, Technical Report 1530, Computer Sciences, University of Wisconsin–Madison, 2006.

[38] A.K. McCallum and K. Nigam, A comparison of event models for naive Bayes text classification, in: *AAAI-98 Workshop on Learning for Text Categorization*, Vol. 752, 1998.

[39] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification*, Wiley, New York, 2001.

[40] R. Aebersold and M. Mann, Mass spectrometry-based proteomics, *Nature* **422**(6928) (2003), 198–207.

[41] D. Lin, D.L. Tabb and J.R. Yates, 3rd, Large-scale protein identification using mass spectrometry, *Biochim. Biophys. Acta* **1646**(1/2) (2003), 1–10.

[42] M. Ganapathiraju, D. Weisser, R. Rosenfeld, J. Carbonell, R. Reddy and J. Klein-Seetharaman, Comparative $n$-gram analysis of whole-genome protein sequences, in: *Proceedings of the HLT'02: Human Language Technologies Conference*, San Diego, CA, March, 2002.

[43] Y. Fofanov, How independent are the appearances of $n$-mers in different genomes?, *Bioinformatics* **20**(15) (2004), 2421–2428.

[44] F. Peng and D. Schuurmans, Combining naive Bayes and $n$-gram language models for text classification, in: *Proceedings of the Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003*, Pisa, Italy, 2003.

[45] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Statist. Soc. Ser. B* **39**(1) (1977), 1–38.

[46] M. Seeger, *Learning with Labeled and Unlabeled Data*, University of Edinburgh, 2001.

[47] E. Segal, Discovering molecular pathways from protein interaction and gene expression data, *Bioinformatics* **19**(90001) (2003), 264–272.

[48] M. Deng, S. Mehta, F. Sun and T. Chen, Inferring domain–domain interactions from protein–protein interactions, *Gen. Res.* **12** (2002), 1540–1548.

[49] G. Bejerano and G. Yona, Variations on probabilistic suffix trees: statistical modeling and prediction of protein families, *Bioinformatics* **17**(1) (2001), 23–43.

[50] C. Cortes and V. Vapnik, Support-vector networks, *Mach. Learn.* **20**(3) (1995), 273–297.

[51] R. Collobert, F. Sinz, J. Weston and L. Bottou, Large scale transductive SVMs, *J. Mach. Learn. Res.* **7** (2006), 1687–1712.

[52] B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. 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 Res.* **31**(1) (2003), 365–370.

[53] W. Li and A. Godzik, Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, *Bioinformatics* **22**(13) (2006), 1658–1659.

[54] Y. Yang, An evaluation of statistical approaches to text categorization, *Inf. Retr.* **1**(1) (1999), 69–90.

[55] Y. Yang and X. Liu, A re-examination of text categorization methods, in: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999, pp. 42–49.

[56] L. Bottou and C.J. Lin, Support vector machine solvers, in: *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste and J. Weston, eds, The MIT Press, Cambridge, MA, 2007, pp. 1–27.

[57] K.J. Park and M. Kanehisa, Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs, *Bioinformatics* **19**(13) (2003), 1656–1663.

[58] A. Garg, M. Bhasin and G.P. Raghava, SVM-based method for subcellular localization of human proteins using amino acid compositions, their order and similarity search, *J. Biol. Chem.* **280**(15) (2005), 14427–14432.

[59] O. Chapelle, V. Vapnik, O. Bousquet and S. Mukherjee, Choosing multiple parameters for support vector machines, *Mach. Learn.* **46**(1/3) (2002), 131–159.

[60] C.S. Yu, Y.C. Chen, C.H. Lu and J.K. Hwang, Prediction of protein subcellular localization, *Proteins* **64**(3) (2006), 643–651.