Assessing the Performance of Macromolecular Sequence Classifiers

Cornelia Caragea, Jivko Sinapov, Vasant Honavar
Computer Science Department
Iowa State University
Ames, Iowa, USA
Email: {cornelia, jsinapov, honavar}@cs.iastate.edu

Drena Dobbs
Department of Genetics and Cell Biology
Iowa State University
Ames, Iowa, USA
Email: ddobbs@iastate.edu

***Abstract*—Machine learning approaches offer some of the most cost-effective approaches to building predictive models (e.g., classifiers) in a broad range of applications in computational biology. Comparing the effectiveness of different algorithms requires reliable procedures for accurately assessing the performance (e.g., accuracy, sensitivity, and specificity) of the resulting predictive classifiers. The difficulty of this task is compounded by the use of different data selection and evaluation procedures and in some cases, even different definitions for the same performance measures. We explore the problem of assessing the performance of predictive classifiers trained on macromolecular sequence data, with an emphasis on cross-validation and data selection methods. Specifically, we compare sequence-based and window-based cross-validation procedures on three sequence-based prediction tasks: identification of glycosylation sites, RNA-Protein interface residues, and Protein-Protein interface residues from amino acid sequence. Our experiments with two representative classifiers (Naive Bayes and Support Vector Machine) show that sequence-based and windows-based cross-validation procedures and data selection methods can yield different estimates of commonly used performance measures such as accuracy, Matthews correlation coefficient and area under the Receiver Operating Characteristic curve. We argue that the performance estimates obtained using sequence-based cross-validation provide more realistic estimates of performance than those obtained using window-based cross-validation.**

I. Introduction

Advances in high throughput data acquisition technologies have resulted in rapid increase in the amount of data, e.g., DNA and protein sequences, protein structures, and gene expression data in biological sciences. Machine learning algorithms offer some of the most cost-effective methods for acquiring useful knowledge and new insights from massive datasets. Recent availability of easy-to-use implementations of machine learning algorithms has led to their increasing use by biologists on a broad range of problems in bioinformatics.

Therefore, there is a growing need for reliable procedures for accurately assessing the performance of predictive models (e.g., classifiers) trained using machine learning algorithms as well as better understanding of the limitations of some of the commonly used evaluation procedures.

In the case of classification tasks, several commonly used performance measures (e.g., accuracy, sensitivity, and specificity, area under the ROC curve) seek to evaluate the quality of the predictions [1]. Each of these measures summarize the information obtained from the estimated numbers of true positives, false positives, true negatives, and false negatives. Unfortunately, there are often inconsistencies in the definition of these performance measures across different papers. Even when the same definitions are used, the procedures used to estimate them can vary significantly across the papers, making it difficult even to compare results on the same dataset. In order for the estimated performance measures to be reliable, the distribution of the training and test sets should correspond to the "natural" distribution of the sequences that is likely to be encountered in the real world applications.

Performance estimates are typically obtained using k-fold cross-validation [2]: The "data set" is divided into k equal parts. In each experiment, $k-1$ of these are used to train the classifier and the remainder is used to evaluate the classifier. The desired performance measure (e.g., accuracy) is obtained by averaging the estimated accuracy across the k cross-validation runs. This approach of separating the test set from the training set makes it possible to distinguish between models that "overfit" (memorize) the training data and models that can discover patterns in the data.

Many sequence classification problems involve assigning a class label to the letter (e.g., amino acid) that appears at each position along the sequence. Examples of such classification tasks include prediction of protein secondary structure from the amino acid protein sequence [3], identifying binding and non-binding residues in a protein sequence [4], [5], identifying residues that undergo different post-translational modifications [6], pronouncing English words [7], fraud detection [8], etc. For example, in RNA-Protein interface prediction, the classifier has to predict the binding afinity to RNA for each amino acid in the protein sequence. Because most machine learning algorithms are designed to work with a fixed number of input features, it is fairly common to use a fixed length window corresponding to the target amino acid and an equal number of its sequence neighbors on each side as input to the classifier. Thus, a sliding window is used to generate a collection of fixed length windows from each sequence. The target residue at the center of the window is labeled with the appropriate class label (e.g., binding or non-binding site) [9]. Now consider the application of cross-validation to assess the performance of a classifier on such a "data set" of labeled fixed length

windows. Because each n-letter window shares $n-1$ elements with the preceding and succeeding windows extracted from a given sequence, random partitioning of the "data set" of labeled windows violates the requirement that the training and evaluation data sets be disjoint - a critical requirement for cross-validation. Under this procedure, which we will refer to as *window-based cross-validation* [10], [11], it is likely that both the training and test sets will contain instances that originate from the same sequence. These instances may exhibit highly sequential correlation [9]. More importantly, because of this correlation, during cross-validation it is likely to result in overly optimistic estimates of the performance measures.

Alternatively, cross-validation can be performed at the sequence level by distributing sequences (instead of windows) into the k disjoint folds. We will refer to this procedure as *sequence-based cross-validation*[4], [5]. Unlike *window-based cross-validation*, this procedure guarantees that no part of the same sequence ends up in both the training and test sets. Prediction servers are most often tested on whole sequences in order to annotate features on newly discovered proteins. In such a case, there would not be any part of the new protein included in the training data for that predictive model. In such a setting, *sequence-based cross-validation* offers a more natural approach to estimating the performance of classifiers trained to annotate each position in a sequence with a class label.

Against this background, we compare *sequence-based* and *window-based cross-validation* procedures on three representative sequence-based prediction tasks: identifying glycosylation sites, RNA-Protein interface residues, and Protein-Protein interface residues from amino acid sequence. Our experiments with two representative classifiers (Naive Bayes and Support Vector Machine) show that sequence-based and windows-based cross-validation procedures and data selection methods can yield different estimates of commonly used performance measures such as accuracy, Matthews correlation coefficient and area under the ROC curve. Our results suggest that window-based cross-validation can significantly over-estimate the performance of the classifiers relative to sequence-based cross validation.

The rest of the paper is organized as follows: Section 2 describes the three datasets we used to perform the experiments; Section 3 presents the two cross-validation approaches; Section 4 briefly introduces the machine learning algorithms applied in this study; Section 5 describes the experiments and results; Section 6 concludes with a summary and discussion of the implications of the results.

II. Datasets

We used three datasets to perform experiments, each corresponding to a macromolecular sequence-based prediction task:

O-GLYCBASE dataset contains experimentally verified glycosylation sites compiled from protein databases and literature. O-GlycBase v6.00 [12] does not contain duplicate identical protein sequences, unless there are conflicts in the glycosylation data. The dataset is available online at http://www.cbs.dtu.dk/databases/OGLYCBASE/.

O-Linked glycosylation is a site-specific biological process. It occurs only on Serine (S) and Threonine (T) protein residues. However, not all of these residues in a protein sequence are actually modified by glycosylation. Therefore, we represent S and T sites experimentally verified to be glycosylation sites as positive instances (+) and those not shown experimentally to be either glycosylation or non-glycosylation sites as negative instances (-). The number of S and T sites that are not known to be glycosylation sites (negative instances) is much larger than the number of sites known to be glycosylation sites (positive instances). In addition, it is highly likely that some of the sites considered as negative instances are instead actual glycosylation sites that have not yet been experimentally discovered to undergo glycosylation. The dataset contains 216 glycoprotein sequences and the total number of S/T sites is 14315.

RNA-Protein Interface dataset, RP147, consists of RNA-binding protein sequences extracted from structures of known RNA-Protein complexes solved by X-ray crystallography in the Protein Data Bank [13]. Proteins with sequence identity greater than 30% or structures with resolution worse than $3.5\mathring{A}$ were removed using PISCES [14]. The resulting dataset has 147 protein sequences. The total number of amino acid residues is 32,324. A target residue was identified as RNA-Protein interface residue using ENTANGLE [15] with the default parameters. The dataset is available online at http://bindr.gdcb.iastate.edu/RNABindR/.

Protein-Protein Interface dataset consists of 42 protein sequences, with sequence identity less than 30%. A target residue was identified as an interface residue if its solvent accessible surface area (ASA) computed in the complex is less than that computed in the monomer by at least $1\mathring{A}^2$ [16]. ASA is computed using DSSP program [17]. The total number of interface and non-interface amino acid residues is 11,554.

In the **RNA-Protein Interface** and **Protein-Protein Interface** prediction tasks, we represent residues identified as interface residues in a protein sequence as positive instances (+) and those not identified as interface residues as negative instances (-).

A. Feature Extraction

Because most machine learning algorithms are designed to work with a fixed number of input features, it is common to use a local window of length $2n+1$, $x = x_{-n}x_{-n+1}\cdots x_{-1}x_0x_1\cdots x_{n-1}x_n$, with each target residue x_0 in the middle and its n neighbor residues, x_i, $i = -n,\cdots,n$, $i \neq 0$, on each side as input to the classifier. $x_i \in \Sigma$, $i = -n,\cdots,n$ and $x \in \Sigma^*$, where Σ represents the 20 amino acid alphabet.

For the glycosylation dataset, a local window is extracted for each S/T glycosylated or non-glycosylated site, $x_0 \in \{S,T\}$. For the other two datasets considered, a local window is extracted for every residue in a protein sequence, $x_0 \in \Sigma$, using the "sliding window" approach [9]. The local windows

TABLE I
NUMBER OF POSITIVE (+) AND NEGATIVE (-) INSTANCES USED IN OUR EXPERIMENTS FOR O-GLYCBASE, RNA-PROTEIN, AND PROTEIN-PROTEIN INTERFACE DATASETS

Dataset	Number of Sequences	Number of + Instances	Number of - Instances
O-GlycBase	216	2168	12147
Protein-RNA	147	4336	27988
Protein-Protein	42	2350	9204

for residues close to N- and C-terminals are filled in with missing values to obtain the same window length.

Table I shows the exact number of positive and negative instances (glycosylation versus non-glycosylation sites and interface versus non-interface residues) for O-GlycBase, RNA-Protein Interface and Protein-Protein Interface datasets.

III. CROSS-VALIDATION PROCEDURE

K-fold cross-validation is an evaluation scheme considered by many authors to be a good method of estimating the *generalization accuracy* of a predictive algorithm (i.e. the accuracy with which the predictive algorithm fits examples in the test set). This evaluation scheme can be described as follows: the original dataset containing m instances is randomly partitioned into k disjoint subsets of approximately equal size, $\approx m/k$. The cross-validation procedure is then performed k different times. During i^{th} run, $i = 1, \cdots, k$, the i^{th} subset is used for testing and the remaining $k-1$ subsets are used for training. Therefore, each instance in the dataset is used exactly once in the test set and $k-1$ times in the training set during the k cross-validation experiments. The results from the k different runs are then averaged. K-fold cross-validation can be repeated several times, each time with a different seed for randomly splitting the dataset. The more k-fold cross-validation is repeated, the lower the variance of the estimates.

In many papers, cross-validation is done in the following way: local windows (instances) are extracted for each residue in a protein sequence, using the sliding window approach, or for some specific residues; similar (within a certain percent of identity) or identical instances are removed from the dataset; the resulting dataset is split into different subsets according to some criteria. Culling datasets to reduce sequence identity helps to avoid overestimation of performance measures [10], [11]. This approach is called *window-based k-fold cross-validation*. We note several problems with this method:

- During k evaluation runs, both the train and test sets are likely to contain some instances that originate from the *same sequence*. Thus, the train and test data sets are *not disjoint* at the sequence level. By using the sliding window approach to extract local window information, two windows corresponding to two consecutive amino acid residues in a sequence differ by exactly one residue, resulting in very high correlation between the two instances. This violates the independence assumption between train and test sets.
- As noted above, in *window-based cross-validation* "similar" or identical sequence windows are often eliminated from the dataset. However, for many machine learning algorithms, residue position matters (see, for example, the definition of the SVM kernel described in the Machine Learning Methods Section). Therefore, removal of similar windows may be problematic.
- Simply eliminating "similar" or identical sequence windows from the dataset perturbs the "natural" distribution of the data extracted from the original sequence dataset. Ideally, the performance of the classifier must be estimated using the "natural" data distribution.

To overcome these problems with *window-based cross-validation*, a different cross-validation approach, called *sequence-based k-fold cross-validation* is considered in other papers [5], [4]. This is done as follows: the original dataset is built so that there is no pairwise sequence identity greater than a certain percentage (in general 25 or 30%). Unlike in *window-based cross-validation* where windows (instances) are distributed into k disjoint sets, in *sequence-based cross-validation*, sequences are distributed into k disjoint sets, and then using the sliding window approach, instances are extracted in each set. This way, all instances belonging to the same sequence end up in the same set, preserving the "natural" distribution of the original sequence dataset (modulo any of the sequence identity cutoff).

IV. MACHINE LEARNING METHODS

A. Support Vector Machine Classifier

Support Vector Machine (SVM) classifier is one of the most effective machine learning algorithms for many complex binary classification problems [18]. It is a supervised learning algorithm that belongs to the class of discriminative models.

Given a set of labelled inputs $(\mathbf{x}_i, y_i)_{i=1,\cdots,l}$, $\mathbf{x}_i \in \mathbf{R}^n$ and $y_i \in \{-1, +1\}$, learning an SVM classifier is equivalent to learning a linear decision function $f(\mathbf{x}) =< \mathbf{x}, \mathbf{w} > +b$, $\mathbf{w} \in \mathbf{R}^n$ and $b \in \mathbf{R}$, that accurately discriminate between positive and negative labelled inputs. This can be achieved by solving a dual quadratic optimization problem. $\mathbf{w}$ and b are optimized such that the "margin" of separation (the distance) between the two classes is maximized.

During classification, an unlabelled input $\mathbf{x}$ is assigned to a class based on the sign of the linear decision function, $sgn(f(\mathbf{x}))$ (e.g., if $f(\mathbf{x}) > 0$ then $\mathbf{x}$ is assigned to the positive class; otherwise $\mathbf{x}$ is assigned to the negative class) [19].

However, for many real-world problems, such as biological problems, a linear decision function $f(\mathbf{x})$ in the n-dimensional input space cannot be learned. In these cases, SVM algorithm works by mapping the labelled inputs into a (possibly) higher-dimensional *feature space* through an appropriate feature map, $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$, $i = 1, \cdots, l$, where a linear decision function can be found. Rather than explicitly computing the feature vector for each data point $\mathbf{x}_i$, the mapping is defined implicitly via a *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j) =< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$, $i, j = 1, \cdots, l$ that satisfies the Mercer's Condition [18]. The *kernel function*

is evaluated for each pair of data points, and it specifies a similarity measure between them.

In this study, we used 0/1 String Kernel with SVM classifier, the mappings being done implicitly. This kernel specifies a similarity measure between two local windows x and y based on their identities. Formally, this is defined as:

$$K(x, y) = (\sum_{i=-n}^{n} I[x_i = y_i])^2 \quad (1)$$

where $I[.]$ is the indicator function ($I[x_i = y_i] = 1$ if the amino acids on the i^{th} position of the two local windows are the same, $x_i = y_i$, and $I[x_i = y_i] = 0$, otherwise). The higher the value of the kernel $K(x, y)$, the more similar the local windows x and y.

B. Naive Bayes Classifier

Naive Bayes classifier [2] is one of the simplest probabilistic approaches. It belongs to the class of generative models, in which the probabilities $p(\mathbf{x}|y)$ and $p(y)$ of the input $\mathbf{x}$ and the class label y are estimated from the data. In general the input $\mathbf{x}$ is high-dimensional, represented as a tuple of attribute values, $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$, making it impossible to estimate $p(\mathbf{x}|y)$ for large values of n. However, Naive Bayes classifier makes the assumption that the attribute values are conditionally independent given the class. Therefore, training a Naive Bayes classifier reduces to estimating probabilities $p(\mathbf{x}_i|y)$, $i = 1, \cdots, n$, and $p(y)$. During classification, Bayes Rule is applied to compute $p(y|\mathbf{x}_{test})$ and the class label with the highest posterior probability is assigned to the new input $\mathbf{x}_{test}$.

V. Experiments and Results

A. Experimental Design

For each classification task considered in this study, we performed experiments using *window-based*, and *sequence-based k-fold cross-validation*. In the former, *k*-fold cross-validation is performed on the set of local windows extracted from all the protein sequences in each dataset. In the latter, *k*-fold cross-validation is performed on the set of protein sequences in each dataset. Thus, protein sequences are randomly distributed into folds and then local windows are extracted from sequences in each fold.

Typically, most biological datasets are highly unbalanced, i.e. the number of negative instances is much larger compared to the number of positive instances. Unbalanced data sets present challenges to most standard machine learning algorithms, such as the ones discussed in this study. When the dataset is unbalanced, the traditional performance measure of accuracy is not a good indicator of the performance of the classifier because the classifier will be biased towards the class with larger number of instances (negative class in our case). In such a setting, even a classifier that always labels instances as negatives would give a reasonably good accuracy while performing unnacceptably poor on the minoirity class (positive class in our case) [20]. To avoid this problem, we optimized the classification threshold θ: an instance is predicted as positive if $P(c = +1|\mathbf{x}) > \theta$ (where $P(c = +1|\mathbf{x})$ is the estimated probability of the positive class given $\mathbf{x}$), and negative otherwise. We selected the best classification threshold θ such that a given performance measure (Matthews correlation coefficient) is optimized on the training data. Note however, that optimizing machine learning classsifier with unbalanced data sets is not the focus of this study. Any of a wide range of methods described in the literature on learning classifiers from unbalanced data could be used to replace the simple method used here [20].

Because Support Vector Machine (SVM) classifiers solve a dual quadratic optimization problem to find the linear decision function, it is computationally expensive to train them when the number of instances in the dataset is very large. As can be seen in Table I, our biological datasets are very large. Thus, we used an *ensemble learning* approach [21] to reduce the overall training time of SVM classifiers. Instead of training a single classifier, we trained a collection of classifiers, with each being trained on a balanced fraction of the data, obtained by randomly sampling from the whole data. We used weighted majority vote to combine their predictions.

We evaluated the effect of dataset size on the performance measures for the two types of cross-validation procedures. Thus, we randomly sampled different fraction of the data, starting with 10% of the sequences, and iteratively increasing the size of the dataset by 2.5% of sequences, until all sequences were used.

A major challenge that one may encounter while doing *sequence-based cross-validation* experiments is that although the k disjoint sets have approximately the same number of sequences, they could be highly unbalanced in terms of both the number of instances in each set if the sequence lengths vary signficantly, and the number of instances in each set belonging to different classes.

B. Performance Evaluation

To assess the performance of our classifiers we reported the following measures described in [1]: Accuracy, Matthews Correlation Coefficient (CC), Sensitivity, and Specificity. If we denote true positives, false negatives, false positives, and true negatives by TP, FN, FP, and TN respectively, then these measures can be defined as follows:

$$\texttt{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \quad (2)$$

$$\texttt{CC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \quad (3)$$

$$\texttt{Sensitivity} = \frac{TP}{TP + FN} \quad (4)$$

$$\texttt{Specificity} = \frac{TP}{TP + FP} \quad (5)$$

TABLE II
EXPERIMENTAL RESULTS FOR WINDOW-BASED K-FOLD CROSS-VALIDATION (WINCV), AND SEQUENCE-BASED K-FOLD CROSS-VALIDATION (SEQCV) FOR O-GLYCBASE DATASET USING A SINGLE NAIVE BAYES AND AN ENSEMBLE OF SUPPORT VECTOR MACHINE (SVM) CLASSIFIERS

Classifier/	Naive Bayes		Ensemble of SVMs	
Performance Measure	WinCV	SeqCV	WinCV	SeqCV
Accuracy	**0.93**	0.86	**0.94**	0.88
MCC	**0.69**	0.54	**0.75**	0.53
Sensitivity	0.55	**0.72**	**0.64**	0.60
Specificity	**0.95**	0.53	**0.96**	0.61
AUC	**0.91**	0.87	**0.93**	0.90

TABLE III
EXPERIMENTAL RESULTS FOR WINDOW-BASED K-FOLD CROSS-VALIDATION (WINCV), AND SEQUENCE-BASED K-FOLD CROSS-VALIDATION (SEQCV) FOR RNA-PROTEIN INTERFACE DATASET USING A SINGLE NAIVE BAYES AND AN ENSEMBLE OF SUPPORT VECTOR MACHINE (SVM) CLASSIFIERS

Classifier/	Naive Bayes		Ensemble of SVMs	
Performance Measure	WinCV	SeqCV	WinCV	SeqCV
Accuracy	**0.83**	**0.83**	**0.82**	**0.82**
MCC	**0.34**	0.30	**0.34**	0.33
Sensitivity	**0.32**	0.18	**0.34**	**0.34**
Specificity	0.58	**0.72**	**0.55**	0.54
AUC	**0.74**	0.73	**0.74**	**0.74**

TABLE IV
EXPERIMENTAL RESULTS FOR WINDOW-BASED K-FOLD CROSS-VALIDATION (WINCV), AND SEQUENCE-BASED K-FOLD CROSS-VALIDATION (SEQCV) FOR PROTEIN-PROTEIN INTERFACE DATASET USING A SINGLE NAIVE BAYES AND AN ENSEMBLE OF SUPPORT VECTOR MACHINE (SVM) CLASSIFIERS

Classifier/	Naive Bayes		Ensemble of SVMs	
Performance Measure	WinCV	SeqCV	WinCV	SeqCV
Accuracy	**0.68**	0.64	**0.70**	**0.70**
MCC	**0.14**	0.13	0.19	**0.20**
Sensitivity	0.42	**0.46**	0.47	**0.48**
Specificity	**0.29**	0.28	**0.33**	**0.33**
AUC	**0.62**	0.61	**0.66**	0.65

The Receiver Operating Characteristic (ROC) curve plots the proportion of correctly classified positive examples, True Positive Rate (TPR) as a function of the proportion of incorrectly classified negative examples, False Positive Rate (FPR) for different classification thresholds. In comparing two different classifiers using ROC curves, for the same False Positive Rate, the classifier with higher True Positive Rate gives better performance measures. Each point on the ROC curve represents a classification threshold θ and corresponds to particular values of TPR and FPR.

To evaluate how good a classifier is at discriminating between the positive and negative examples, we also report the Area Under the ROC Curve (AUC) on the test set, which represents the probability of correct classification ([22], [1]). That is, an AUC of 0.5 indicates a random discrimination between positives and negatives (random classifier), while an AUC of 1 indicates a perfect dicrimination (very good classifier).

C. Results

For each classification task considered in this study, we trained and evaluated single Naive Bayes and ensemble of Support Vector Machine (SVM) classifiers, using both *window-based* and *sequence-based k-fold cross-validation* (we set $k = 10$ in our experiments). To reduce the variance of the estimated performance measures, we repeated k-fold cross-validation 50 times for each Naive Bayes experiment. We computed the performance measures by averaging the results obtained after every k-fold cross-validation run. Due to the large number of instances in each dataset and the time complexity of training SVM classifiers, we performed k-fold cross-validation only

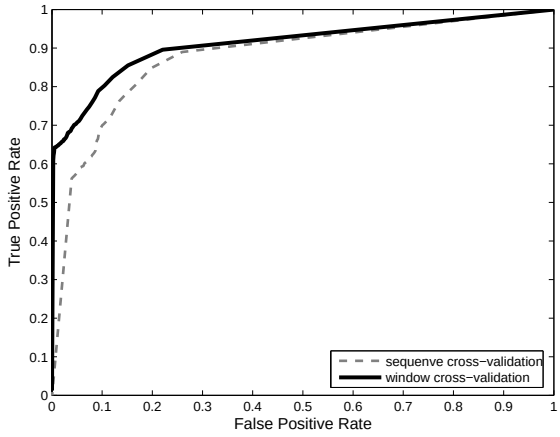

Fig. 1. Receiver Operaring Characteristic (ROC) curves for window-based and sequence-based cross-validation experiments using an ensemble of SVM classifiers on O-GlycBase dataset

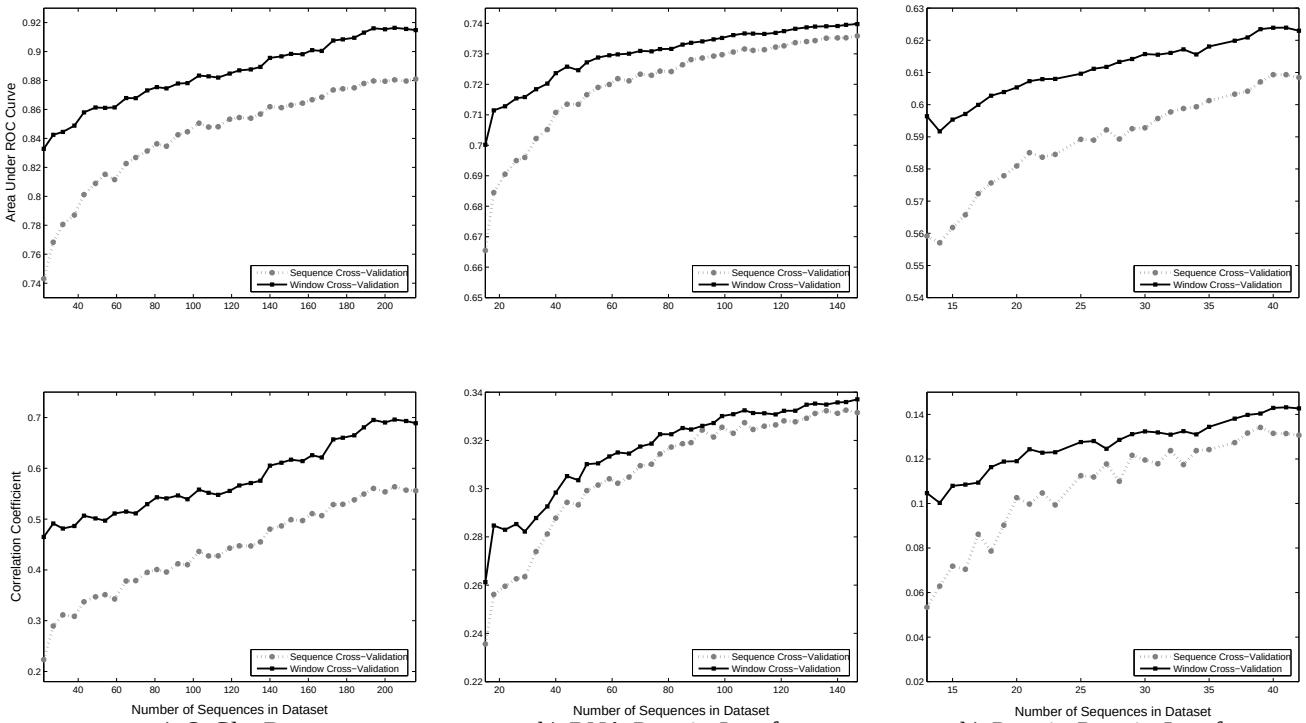

Fig. 2. Comparison of Area Under the ROC Curve (AUC) (upper plots) and Matthews Correlation Coefficient (lower plots) between window-based and sequence-based cross-validation on O-GlycBase (left), RNA-Protein Interface (middle), and Protein-Protein Interface (right) datasets from which we randomly sampled different fraction of the data, starting with 10% of the sequences, and iteratively increasing the size of the data by 2.5% of sequences, until all sequences were used.

once for each SVM experiment. The kernel for SVM classifier is 0/1 String Kernel, as described in the Machine Learning Methods section. In all our experiments, we used a window size of n=21 (10 residues on each side of the target residue) for each instance.

We show experimental results in Tables II - IV. As can be seen, in most cases, the performance measures obtained with *window-based k-fold cross-validation* are more optimistic than those obtained with *sequence-based k-fold cross-validation*. For example, Area Under the ROC Curve (AUC) is always larger using *window-based k-fold cross-validation* for any of the three classification tasks considered. Matthews Correlation Coefficient (CC) is larger using *window-based* than *sequence-based k-fold cross-validation* on all three classification tasks using a single Naive Bayes, and two classification tasks using an ensemble of SVMs.

In Figure 1, we compare the ROC curves for *window-based* and *sequence-based cross-validation* experiments using an ensemble of SVMs on O-GlycBase dataset. It can be seen that *window-based cross-validation* has a larger AUC than *sequence-based cross-validation*.

These results seem to validate the assertion that *window-based k-fold cross-validation* overestimates the performance of a classifier relative to that obtained using *sequence-based k-fold cross-validation*.

We also trained and evaluated single Naive Bayes on different fractions of the data that were randomly sampled from the whole data (as explained in the Experimental Design section), using both *window-based* and *sequence-based k-fold cross-validation*. In Figure 2, we compare AUC and CC obtained using *window-based* and *sequence-based cross-validation* for all three classification tasks when we iteratively increase the size of the dataset. These results show that the difference in performance measures is higher when the data are scarce, and becomes smaller as more data are used. Eventually, the performance measures for the two approaches tend to converge. However, it is unclear how much data would be needed in order for them to converge to a single point. These results suggest that as the dataset grows in size, the dependencies between the training and test sets that are present in window-based cross-validation become less and less of an advantage for the classifier.

VI. Conclusion

With the increasing use of machine learning approaches in bioinformatics, there is a growing need for reliable procedures for accurately assessing the performance of predictive models (e.g., classifiers) trained using machine learning algorithms as well as better understanding of the limitations of some of the commonly used evaluation procedures.

In this paper, we focus on some of the problems of assessing the performance of classifiers trained on macromolecular sequence data, with an emphasis on cross-validation and data selection methods. Specifically, we compare two variants of k-fold cross-validation in this setting: *window-based k-fold cross-validation*, in which windows are distributed randomly into k disjoint sets; and *sequence-based k-fold cross-validation*, in which sequences are distributed into k disjoint sets.

Several datasets drawn from representative sequence-based prediction tasks were used in this study: identifying glycosylation sites, RNA-Protein interface residues, and Protein-Protein interface residues from amino acid sequence. Our experiments with two representative classifiers (Naive Bayes and Support Vector Machine) show that *sequence-based* and *windows-based cross-validation* procedures and data selection methods can yield different estimates of commonly used performance measures such as accuracy, correlation coefficient and area under the ROC curve.

Our results suggest that window-based cross-validation can yield overly optimistic estimates of the performance of the classifiers relative to the estimates obtained using sequence-based cross-validation. The smaller the size of the training dataset, the greater is the magnitude of the difference between the estimates obtained using the two procedures. Also, the effect is more significant for specific types of datasets. For example, on the glycosylation site prediction task, *windows-based cross-validation* yields significantly more optimistic estimates of classifier performance than those obtained using *sequence-based cross-validation*. However, the differences in the performance estimates obtained using the *windows-based* and *sequence-based cross-validation* are smaller in the case of RNA-Protein and Protein-Protein interface residue prediction. *Because predictors trained on labeled sequence data have to predict the labels for residues in a novel sequence, we believe that the estimates obtained using sequence-based cross-validation provide more realistic estimates of performance than those obtained using window-based cross-validation.*

Additional challenges are presented both in the design of machine learning algorithms and the assessment of performance of trained classifiers by the imbalances in the distribution of class labels (as is typical in the case of many macromolecular sequence-based prediction problems), the differences in the distribution of lengths of the sequences, and heterogeneity of sequences in the dataset (e.g., arising from differences in sequences associated with different evolutionary lineages, functional or structural families, etc.). Thus, there is a need for more systematic theoretical as well as empirical studies of the underlying classification problems as well as performance assessment procedures.

ACKNOWLEDGMENT

This research has been supported in part by a grant from the National Institutes of Health (GM066387) to Vasant Honavar and Drena Dobbs.

REFERENCES

[1] P. Baldi, S. Brunak, Y. Chauvin, C. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[2] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[3] N. Qian and T. Sejnowski, "Predicting the secondary structure of globular proteins using neural networks models," *J. Molecular Biology*, vol. 202, pp. 865–884, 1988.

[4] C. Yan, M. Terribilini, R. Wu, F.and Jernigan, D. Dobbs, and V. Honavar, "Identifying amino acid residues involved in protein-dna interactions from sequence," *BMC Bioinformatics*, 2006.

[5] M. Terribilini, J.-H. Lee, C. Yan, R. L. Jernigan, V. Honavar, and D. Dobbs, "Predicting rna-binding sites from amino acid sequence," *RNA Journal*, vol. In Press, 2006.

[6] N. Blom, T. Sicheritz-Ponten, R. Gupt, S. Gammeltoft, and S. Brunak, "Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence," *Proteomics*, vol. 4, no. 6, pp. 1633–49, 2004.

[7] T. Sejnowski and C. Rosenberg, "Parallel networks that learn to pronounce english text." *Journal of Complex Systems*, vol. 1, no. 1, pp. 145–168, 1987.

[8] T. Fawcett and F. Provost, "Adaptive fraud detection," *Knowledge Discovery and Data Mining*, vol. 1, pp. 291–316, 1997.

[9] T. G. Dietterich, "Machine learning for sequential data: A review." *Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science*, vol. 2396, pp. 15–30, 2002.

[10] J. H. Kim, J. Lee, B. Oh, K. Kimm, and I. Koh, "Prediction of phosphorylation sites using SVMs," *Bioinformatics*, vol. 20, no. 17, pp. 3179–3184, 2004.

[11] S. Li, B. Liu, R. Zeng, Y. Cai, and Y. Li, "Predicting o-glycosylation sites in mammalian proteins by using svms." *Comput Biol Chem*, vol. 30, no. 3, pp. 203–8, 2006.

[12] R. Gupta, H. Birch, K. Rapacki, S. Brunak, and J. Hansen, "O-glycbase version 4.0: a revised database of o-glycosylated proteins," *Nucleic Acids Res.*, vol. 27, no. 1, pp. 370–2, 1999.

[13] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne, "The protein data bank," *Nucleic Acid Res*, vol. 28, pp. 235–242, 2000.

[14] G. Wang and R. Dunbrack, "Pisces:a protein sequence culling server," *Bioinformatics*, vol. 19, pp. 1589–1591, 2003.

[15] J. Allers and Y. Shamoo, "Structure-based analysis of protein-rna interactions using the program entangle," *J mol Biol*, vol. 311, pp. 75–86, 2001.

[16] S. Jones and J. Thornton, "Principles of protein-protein interactions," *Proc. Natl Acad. Sci, USA*, vol. 93, pp. 13–20, 1996.

[17] W. Kabsch and C. Sander, *Biopolymers*, vol. 22, p. 2577, 1983.

[18] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[19] V. Vapnik, *Statistical learning theory*. Springer-Verlag, New York, 1998.

[20] M. C. Monard and G. E. A. P. A. Batista, "Learning with Skewed Class Distributions," in *Advances in Logic, Artificial Intelligence and Robotics*, J. M. Abe and J. I. da Silva Filho, Eds. São Paulo, SP: IOS Press, 2002, pp. 173–180.

[21] T. G. Dietterich, "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, vol. 1857, pp. 1–15, 2000.

[22] M. Gribskov and N. Robinson, "Th use of receiver operating characteristic (roc) analysis to evaluate sequence matching," 1996.