Probabilistic Prediction of Protein Phosphorylation Sites Using Classification Relevance Units Machines

Mark Menor, Kyungim Baek, and Guylaine Poisson
Department of Information and Computer Sciences, University of Hawai`I at Mānoa
1680 East-West Road, Honolulu, HI 96822, USA
{mmenor, kyungim, guylaine}@hawaii.edu

ABSTRACT

Phosphorylation is an important post-translational modification of proteins that is essential to the regulation of many cellular processes. Although most of the phosphorylation sites discovered in protein sequences have been identified experimentally, the *in vivo* and *in vitro* discovery of the sites is an expensive, time-consuming and laborious task. Therefore, the development of computational methods for prediction of protein phosphorylation sites has drawn considerable attention. In this work, we present a kernel-based probabilistic Classification Relevance Units Machine (CRUM) for *in silico* phosphorylation site prediction. In comparison with the popular Support Vector Machine (SVM) CRUM shows comparable predictive performance and yet provides a more parsimonious model. This is desirable since it leads to a reduction in prediction run-time, which is important in predictions on large-scale data. Furthermore, the CRUM training algorithm has lower run-time and memory complexity and has a simpler parameter selection scheme than the Relevance Vector Machine (RVM) learning algorithm.

To further investigate the viability of using CRUM in phosphorylation site prediction, we construct multiple CRUM predictors using different combinations of three phosphorylation site features – BLOSUM encoding, disorder, and amino acid composition. The predictors are evaluated through cross-validation and the results show that CRUM with BLOSUM feature is among the best performing CRUM predictors in both cross-validation and benchmark experiments. A comparative study with existing prediction tools in an independent benchmark experiment suggests possible direction for further improving the predictive performance of CRUM predictors.[1]

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: Biology and Genetics; I.5 **[Pattern Recognition]**: Models; I.5.1 **[Models]**: Statistical

General Terms

Algorithms, Experimentation, Performance

Keywords

Phosphorylation, Classification, Kernel machine

[1]This work is based on an earlier work: SAC '12 Proceedings of the 2012 ACM Symposium on Applied Computing, Copyright 2012 ACM 978-1-4503-0857-1/12/03. http://doi.acm.org/10.1145/2245276.2231997.

1. INTRODUCTION

Phosphorylation is one of the most widespread post-translational modifications of proteins, and it plays an essential role in the regulation of many cellular processes, such as metabolism, immune response, apoptosis, and cellular motility [39]. Phosphorylation occurs when a protein kinase (PK) enzyme covalently bonds phosphate groups to particular amino acids in a protein [42]. Phosphorylation is reversible through a process called dephosphorylation, where phosphatase enzymes remove the attached phosphate groups. Typically, one of these states, phosphorylated or unphosphorylated, set the protein to be biochemically active, while the other state leaves the protein inactive. Determining which sites of a protein is phosphorylatable is thus important in the understanding of the regulation of biochemical processes, particularly of diseases, making PKs a major target for drug design [7]. For example, a drug called *imatinib* is used to treat some cancers via inhibiting an associated PK, preventing the cancer's anti-apoptopic processes from being activated via phosphorylation [13].

There is a variety of ways to discover protein phosphorylation sites experimentally, and the mass spectrometry approach has been one of the most popular in recent years [21]. The experimentally verified phosphorylation sites are listed in the annotations in a variety of protein databases, including Swiss-Prot [5], Phospho-Base [23], and PhosphoSite [17]. Regardless of the method used, however, experimental discovery of phosphorylation sites is an expensive and laborious task with many technical challenges [39]. To help mitigate this problem, computational approaches that identify highly probable phosphorylation sites have been an active area of research for over ten years. As a result, numerous computational techniques have been proposed [39, 42] with the focus on predicting phosphorylatable serine (S), threonine (T), and tyrosine (Y), as these are the types of acceptor residues dominating the databases. Often, biologists apply these methods to narrow down the list of possible phosphorylation sites to be experimentally verified in the proteins that they are studying [39].

The majority of existing phosphorylation site prediction tools uses methods from supervised learning. These include Support Vector Machine (SVM) used by KinasePhos [41], Musite [14], and PPRED [2], and a variety of neural network methods used by NetPhosK [4] and GANNPhos [36], among others. These tools differ in their choice of site features and encodings, and their choice of learning algorithms. Moreover, these tools can be grouped into two categories: kinase-independent and kinase-specific. A kinase-independent predictor predicts whether a

protein site is phosphorylatable or not, regardless of what PK catalyzed the site. The aforementioned PPRED is an example of such a predictor. Another example is DISPHOS [18] that employs logistic regression to distinguish phosphorylatable sites using information on the predicted 3D structure of the protein. Specifically, DISPHOS uses computational disorder predictors to determine whether a protein site is in an intrinsically disordered protein region that does not fold into a stable 3D structure. Such protein structure information is useful, as phosphorylation sites are frequently contained in disordered protein regions [18].

In addition to predicting whether a site is phosphorylatable or not, a kinase-specific predictor also estimates a candidate PK or type of PK that could cause the phosphorylation. This is particularly important for phosphorylation sites discovered using the popular mass spectrometry method, as the method does not identify the PK that catalyzes the site [39]. To identify the PK, the predictor NetPhosK [4] uses Mutilayer Perceptrons on sequence information to predict 17 common PKs. Due to the large number of PKs (estimated to be greater than 500 in mammals) and very little experimental data on the majority of them, many predictors instead focus on predicting PK groups or families, rather than individual PKs [42]. Predictors like KinasePhos [41] subdivide the large kinase groups into smaller, more specific groups. Using Maximum Dependence Decomposition to subdivide the large PK groups into smaller groups, KinasePhos learns SVMs on sequence information to make predictions from about 60 PK subgroups.

In this study, we propose a kernel-based learning method, the Classification Relevance Units Machine (CRUM), and present its application to prediction of protein phosphorylation sites. CRUM predictors are kinase-independent, and make probabilistic predictions of phosphorylation sites. Experimental results show that CRUM produces a simpler and yet effective prediction system.

In the next section, we briefly describe the two kernel machines for classification that have been extensively studied in the machine learning community – SVM and RVM. We note that RVM as well as CRUM have not been previously used to solve the phosphorylation site prediction problem. Detailed description of our newly proposed CRUM learning model and algorithm follows in Section 3. Experimental setup including dataset construction and performance assessment measures is described in Section 4, which is followed by presentation of results and analysis of three experiments conducted to assess the effectiveness and the viability of using CRUM in phosphorylation site prediction. Finally, we conclude our work with summary and future prospects of overcoming the limitation and further improving predictive performance of CRUM predictors.

2. KERNEL MACHINES

We assume that the data is a set of N d-dimensional vectors $X = \{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_N\} \subset \mathbf{R}^d$, along with their corresponding target values $\mathbf{t} = [t_1, t_2,\ldots, t_N]^T$, where $(\mathbf{x}_i, t_i)$ are drawn independently and identically from a distribution. Since we are only considering binary classification, $t_i \in \{0, 1\}$ for the CRUM and RVM, and $t_i \in \{-1, 1\}$ for the SVM. The value of t_i is set to 1 if site $\mathbf{x}_i$ is phosphorylatable, otherwise t_i is set to 0 or –1 for CRUM/RVM and SVM, respectively.

2.1 Support Vector Machine

The goal of an SVM is to find the hyperplane decision boundary that perfectly separates the two classes in the training dataset with maximum margin [40]. This means the distance from the closest training point to the hyperplane should be maximized. The SVM has been generalized to allow for overlapping class distributions [10] and non-linear decision boundaries via use of kernel functions [6]. A commonly used kernel function, and the one used in this study, is the Gaussian radial basis function (RBF) kernel with parameter $\gamma > 0$:

$$k(\mathbf{x}, \mathbf{x}') = exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2). \quad (1)$$

To use the SVM, a user must specify values for the parameter C that controls the trade-off between minimizing training error and model complexity, and all the kernel parameters; namely, γ in the case of the Gaussian RBF kernel.

Upon completion of training, we can classify a new data point $\mathbf{x}$ using the following formula,

$$\hat{t}(\mathbf{x}) = sgn\left(\sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \quad (2)$$

where the w_i's and b are learned through the SVM training [6] and $sgn(x)$ is the sign function that evaluates to –1 if $x < 0$ and 1 if $x > 0$. As a consequence of the SVM training, many of the w_i's will vanish. Therefore only a subset of the training data is required to make predictions. The training data $\mathbf{x}_i$ whose associated $w_i > 0$ are called support vectors (SVs).

2.2 Relevance Vector Machine

Unlike the SVM, the RVM seeks to model the posterior distribution $p(C_+|\mathbf{x})$ that a site $\mathbf{x}$ is phosphorylatable (i.e. is a member of the positive class C_+) using the following model,

$$p(C_+ \mid \mathbf{x}) = \hat{t}(\mathbf{x}) = \sigma\left(\sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \quad (3)$$

where $\sigma(t) = (1 + e^{-t})^{-1}$, and the w_i's and b are learned through the RVM training. Empirically it has been seen that more w_i terms vanish with the RVM than with the SVM, resulting in a sparser, more parsimonious model [37]. Similar to the idea of SVs, the vectors $\mathbf{x}_i$ whose associated $w_i > 0$ are called relevance vectors (RVs). While there has been an addition to the SVM to allow for probabilistic outputs [28], it has been shown that SVMs make poor estimates of the posterior probability due to the nature of its objective function [37]. To use the RVM, the user needs to only specify values for the kernel parameters.

3. CLASSIFICATION RELEVANCE UNITS MACHINE

We propose a new classification model CRUM, which combines a sparse kernel regression model called the Relevance Units Machine [15] with the logistic regression model [16]. To obtain probabilistic predictions, the CRUM seeks to model the posterior distribution $p(C_+|\mathbf{x})$ that a site $\mathbf{x}$ is phosphorylatable (i.e. is a member of the positive class C_+) using the following model,

$$p(C_+ \mid \mathbf{x}) = \hat{t}(\mathbf{x}) = \sigma\left(\sum_{i=1}^{M} w_i k(\mathbf{x}, \mathbf{u}_i) + b\right) \quad (4)$$

where M is a positive integer, and the w_i's, b, kernel parameters, and the relevance units (RUs) $\mathbf{u}_i$'s are learned through the CRUM training. A commonly used kernel function, and the one used in this study, is the Gaussian kernel described in Section 2.1. The CRUM can be visualized as a feed-forward network, as shown in Figure 1.

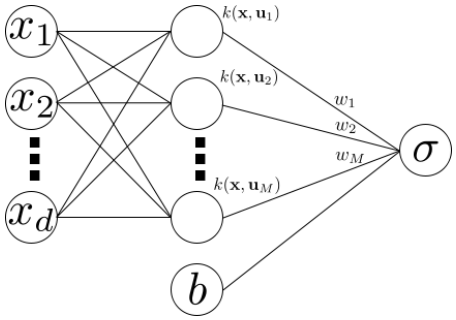

Figure 1. The CRUM model displayed as a feed-forward network. The input layer consists of *d* nodes representing the *d*-dimensional input vector x. The hidden layer consists of bias value *b* and the *M* kernel functions centered on the RUs $\mathbf{u}_i$. Finally, a weighted sum of the hidden layer outputs is computed and transformed into a posterior probability via sigmoid function σ.

Like the RVM, the CRUM provides estimates for the posterior distribution. However, the CRUM training algorithm has lower run-time and memory complexity than the RVM algorithm, requiring the storage of an $N \times M$ rather than an $N \times N$ design matrix [25]. To use the CRUM, the user needs to specify only the model complexity parameter M, typically with $M << N$ to achieve a parsimonious model. A simple scheme for selecting M is proposed later in this section to eliminate the need for costly cross-validation for model selection.

3.1 Overview of Learning Algorithm

Our original learning algorithm described in [25] implements a regularized supervised learning method using approximate Bayesian techniques to learn the full set of model parameters: the coefficients w_i, the bias b, the kernel parameters, and the RUs $\mathbf{u}_i$. However, this full supervised training algorithm is not practical for training on the large phosphorylation datasets due to the high complexity in learning the RUs and kernel parameters. Instead, we propose to learn the model parameters in two phases.

Phase 1 – Clustering: In the first phase, unsupervised learning methods are used to learn the RUs and kernel parameters. For computational simplicity, the k-means clustering algorithm [24] is used to find the M clusters. Since we are using the Gaussian kernel under the Euclidean norm, we use Euclidean distance as the similarity metric for the k-means clustering. Then, the RUs are set to the M cluster centers, and γ is set to $(2r^2)^{-1}$ where r is the maximum distance between cluster centers.

This clustering phase also provides an avenue to select empirically an appropriate value of M. There has been work in the clustering research community for developing methods to determine the number of clusters, such as [35]. There are also clustering algorithms using Bayesian techniques to find the appropriate number of clusters, such as the Variational Bayesian Expectation-Maximization algorithm for Mixture of Gaussians [9]. However, these methods are computationally complex, and we opted to use a simpler method to determine M, as described later. Due to the regularization scheme presented in the next phase, an overestimate of M will not result in overfitting.

Phase 2 – Supervised Learning: Like the original CRUM learning algorithm [25], the w_i's and b are still learned by using the iteratively reweighted least squares (IRLS) algorithm and the hyperparameter α is learned by using type-II maximum likelihood. The hyperparameter α serves as a regularizer, and prevents overfitting caused by overestimating the model complexity M. What follows is a detailed description of Phase 2 of the CRUM learning algorithm.

3.2 Learning Algorithm Assumptions and Derivation

Without loss of generality, we assume $b = 0$ and set $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$ for a fixed positive integer M. Define

$$p(C_+ \mid \mathbf{x}) = \hat{t}(\mathbf{x}) = \sigma\left(\sum_{i=1}^{M} w_i k(\mathbf{x}, \mathbf{u}_i) + b\right) \quad (5)$$

and the kernel matrix $\mathbf{K}$ with elements

$$K_{ij} = k(\mathbf{x}_i, \mathbf{u}_j) \quad (6)$$

for $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}$. Here, the kernel function k is the Gaussian kernel with parameter γ under the Euclidean norm in Equation (1). The RUs $\mathbf{u}_j$'s and γ are fixed to the values learned in Phase 1.

Let $\hat{t}_i = \hat{t}(\mathbf{x}_i)$. We adopt a Bernoulli distribution for $P(\mathbf{t}|\mathbf{x}, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta})$, where $\mathbf{U}$ is the matrix containing all RUs column-wise and $\boldsymbol{\Theta}$ is the set of all kernel parameters, in this case $\boldsymbol{\Theta} = \{\gamma\}$. This leads to the following likelihood given the model:

$$P(\mathbf{t} \mid X, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta}) = \prod_{n=1}^{N} \hat{t}_n^{t_n} \left[1 - \hat{t}_n\right]^{1-t_n}. \quad (7)$$

To mitigate overfitting by reducing the classifier's complexity, we adopt a zero-mean isotropic Gaussian prior distribution over $\mathbf{w}$,

$$p(\mathbf{w} \mid \alpha) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}) \quad (8)$$

where $\mathbf{I}$ is the $M \times M$ identity matrix and $\alpha \in \mathbf{R}^+$ is the hyperparameter controlling the precision of the Gaussian over the weights. Finally, we assume a Gamma distributed hyperprior for the hyperparameter α,

$$p(\alpha) = \text{Gamma}(\alpha|a, b) \quad (9)$$

where Gamma($x|a, b$) = $\Gamma(a)^{-1}b^a x^{a-1}e^{-bx}$ and $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$.

The a and b parameters are fixed to small values to make the hyperpriors non-informative [37].

We obtain the posterior distribution by taking the product of all our assumptions (7) – (9),

$$p(\mathbf{t}, \mathbf{w}, \alpha \mid X, \mathbf{U}, \boldsymbol{\Theta}) = P(\mathbf{t} \mid X, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta})\, p(\mathbf{w} \mid \alpha)\, p(\alpha). \quad (10)$$

The dependencies between all the variables can be visualized using a directed graph called a Bayesian network, as shown in Figure 2.

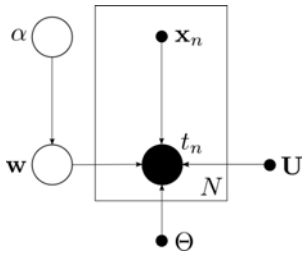

Figure 2. The CRUM model displayed as a Bayesian network. Note that the nodes contained in the center rectangle are repeated *N* times, one for each training data point. When using the Gaussian kernel, Θ = {γ}.

Taking the negative log of Equation (10) and dropping all terms not involving **w** give the following objective function,

$$L(\mathbf{w}) = -\sum_{n=1}^{N}\left[t_n \ln \hat{t}_n + (1-t_n)\ln(1-\hat{t}_n)\right] + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}. \quad (11)$$

The negative summation term in Equation (11) is known as the empirical risk under the log loss, and is an empirical measure of the error of the classifier. The last term of the equation is the penalty function, penalizing complex models over simpler ones. It is known that empirical risk is a biased estimate of the true risk and thus the additional penalty term is required to avoid overfitting [32]. Therefore, the parameter α controls the trade-off between empirical error and model complexity. Thus, minimizing the objective function (11) corresponds to a structural risk minimization [22] and aims to minimize the true error bound. Furthermore, log loss is a margin maximizing loss function [31] and this fact helps to tighten the true error bound to further reduce overfitting [32].

To estimate **w** over a fixed α, we minimize Equation (11), or equivalently maximize Equation (10), with respect to **w**. We can compute the **w** giving the minimum of Equation (11), call it **w***, using the Newton-Raphson method, which is also known in this classification context as the IRLS algorithm. The use of IRLS requires the computation of the gradient and Hessian as follows,

$$\nabla L = \mathbf{K}^T(\hat{\mathbf{t}} - \mathbf{t}) + \alpha\mathbf{w} \quad (12)$$

$$\mathbf{H} = \mathbf{K}^T\mathbf{B}\mathbf{K} + \alpha\mathbf{I} \quad (13)$$

where $\mathbf{B} = diag(\hat{t}_1(1-\hat{t}_1), \hat{t}_2(1-\hat{t}_2), \ldots, \hat{t}_N(1-\hat{t}_N))$ and $\hat{\mathbf{t}} = [\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N]^T$. The update formula for **w** is then

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \lambda\mathbf{H}^{-1}\nabla L \quad (14)$$

for an appropriately chosen step size λ. Pseudocode for learning **w** is presented in Algorithm 1.

Note that the inverse Hessian computed in line 8 can be done efficiently using the Cholesky decomposition [27]. The inverse Hessian is also used in the learning of α; therefore, in addition to returning the optimal **w**, an implementation should also return $\mathbf{H}^{-1}$.

Algorithm 1: Learning **w**

```
 1: w ← initial value
 2: λ_min ← a small value (e.g. 2^−8)
 3: while w has not converged do
 4:     w^old ← w
 5:     L^old ← Equation (11)
 6:     L ← ∞
 7:     ∇L ← Equation (12)
 8:     H ← Equation (13)
 9:     ∇ ← H^−1 ∇L
10:     λ ← 1
11:     {Comment: Line search to find an appropriate λ value}
12:     while λ > λ_min and L > L^old do
13:         w ← w^old − λ∇
14:         L ← Equation (11) using updated w
15:         λ ← λ/2
16:     end while
17:     {Comment: If line search failed, revert to old w value}
18:     if λ < λ_min then
19:         w ← w^old
20:     end if
21: end while
```

To learn α, we use an approximate Bayesian estimate called evidence approximation or type-II maximum likelihood. We marginalize **w** in the posterior (10) and use the Laplace approximation [1],

$$\begin{aligned} p(\mathbf{t}, \alpha \mid X, \mathbf{U}, \boldsymbol{\Theta}) &= \int P(\mathbf{t} \mid X, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta})\, p(\mathbf{w} \mid \alpha)\, p(\alpha) d\mathbf{w} \\ &\approx (2\pi)^{M/2} \mid \boldsymbol{\Sigma} \mid^{1/2} P(\mathbf{t} \mid X, \mathbf{U}, \mathbf{w}^*, \boldsymbol{\Theta})\, p(\mathbf{w}^* \mid \alpha)\, p(\alpha) \end{aligned} \quad (15)$$

where the covariance of the Laplace approximation is

$$\boldsymbol{\Sigma} = (\mathbf{K}^T\mathbf{B}\mathbf{K} + \alpha\mathbf{I})^{-1} \quad (16)$$

and its mean $\mathbf{w}^*$ is the **w** that maximizes $P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta})p(\mathbf{w}|\alpha)$. Note that $\mathbf{w}^*$ and $\boldsymbol{\Sigma}$ are the optimal **w** and the inverse Hessian $\mathbf{H}^{-1}$, respectively, as computed in Algorithm 1.

Taking the derivative of Equation (15) and setting it to zero give the following iterative estimate for α,

$$\alpha^{new} = \frac{M - \alpha^{old}\,\mathrm{tr}[\boldsymbol{\Sigma}]}{\mathbf{w}^{*T}\,\mathbf{w}^{*}}. \quad (17)$$

In summary, the pseudocode for Phase 2 of the CRUM learning algorithm is presented in Algorithm 2.

Note that if we wish, b may be learned by augmenting the weight vector with b, such that $\tilde{\mathbf{w}} = [w_1,\ldots, w_M, b]$. The kernel matrix must also be augmented to include an additional column of ones. Denoting the column vector of ones with **1**, we define the augmented kernel matrix as $\tilde{\mathbf{K}} = [\mathbf{K}, \mathbf{1}]$. Then, we use the aforementioned algorithms using $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{K}}$ instead of **w** and **K**, respectively.

Algorithm 2: CRUM Supervised Learning Algorithm (Phase 2)

1: **w**, $\alpha \leftarrow$ initial values
2: Compute design matrix **K** using Equation (6)
3: **while w** and α have not converged **do**
4: **w** $\leftarrow$ Algorithm 1 using the current value of **w** as initial value
5: $\alpha \leftarrow$ Equation (17)
6: **end while**

3.3 Selecting Model Complexity

Both phases of the CRUM learning algorithm require the specification of a fixed parameter M that represents the number of RUs and thus is a model complexity parameter. Cross-validation could be used by repeatedly using the entire learning algorithm on a variety of selected M values and choosing the model with the best performance. Since this is a slow and time consuming process, we propose a more efficient method that exploits only Phase 1 of the learning algorithm and avoids the costly Phase 2.

During Phase 1, we cluster the unlabeled data X into M clusters using k-means clustering. Giving the clustering result a probabilistic interpretation, the AIC and BIC can be computed [29]. Assume all clusters are spherical Gaussian distributions with identical variance, then the maximum likelihood estimate for the variance is

$$\hat{\sigma}^2 = \frac{1}{N-M}\sum_{i=1}^{N}\left\|\mathbf{x}_i - \boldsymbol{\mu}_{(i)}\right\|^2 \quad (18)$$

where $\boldsymbol{\mu}_{(i)}$ is the closest cluster center to $\mathbf{x}_i$ using Euclidean distance. The log-likelihood of X given the learned clustering distribution $\mathcal{D}_M$ is then

$$l(X \mid \mathcal{D}_M) = \sum_{i=1}^{N}\left(\log\frac{1}{\sqrt{2\pi}\hat{\sigma}^d} - \frac{1}{2\hat{\sigma}^2}\left\|\mathbf{x}_i - \boldsymbol{\mu}_{(i)}\right\|^2 + \log\frac{N_{(i)}}{N}\right) \quad (19)$$

where $N_{(i)}$ is the number of datapoints belonging to the same cluster as $\mathbf{x}_i$. There are $M + Md$ free parameters in this model: M – 1 mixture probabilities, the Md cluster center coordinates, and the variance.

With the log-likehood of the data and the number of free parameters, we can compute the AIC and BIC of the M cluster model using

$$\mathrm{AIC}(M) = 2(Md + M - l(X|\,\mathcal{D}_M)) \quad (20)$$

$$\mathrm{BIC}(M) = (Md + M)\log N - 2l(X|\,\mathcal{D}_M)) \quad (21)$$

where a low AIC or BIC implies a good model. This allows CRUM's M to be selected as the model with the best observed AIC or BIC value. This has clear computational advantages: i) the costly supervised learning algorithm (Algorithm 2) is not invoked and ii) only a single clustering needs to be conducted per M value evaluated, compared to a n-fold cross-validation that would require clustering and supervised training on n different datasets.

4. EXPERIMENTAL SETUP

4.1 Dataset Preparation

We prepared three datasets, both of which are derived from the Phospho.ELM database [12, 11] of known, experimentally verified eukaryotic phosphorylatable S, T, and Y sites. The composition of each dataset is summarized in Table 1.

- *Evaluation dataset*: This dataset contains all known sites from 2009 release of Phospho.ELM.
- *Benchmark test dataset*: The independent dataset taken from [2] was used as benchmark test dataset in this study. This dataset is originally from [33], where it was used to assess the performance of existing predictors. The annotations of this dataset are updated based on Release 9.0 of the Phospho.ELM database [11].
- *Benchmark training dataset*: This dataset contains all known phosphorylation sites from the Phospho.ELM (Release 8.1) database minus the overlapping entries with the benchmark test dataset, providing a training dataset independent of the benchmark test dataset. This dataset was also taken from [2].

Table 1. Composition of datatsets. (#Pos – the number of phosphorylation sites (positives), #Neg – the number of putative non-phosphorylation sites (negatives)).

Dataset	Type	#Pos	#Neg	Comments
Evaluation	S	9129	10137	Used in the comparison of learning methods
	Y	2206	2799	
	T	1555	1691	
Benchmark Test	S	1255	15459	Test dataset for benchmark experiment
	T	353	9995	
	Y	396	4795	
Benchmark Training	S	8000	8000	Used in 1) cross-validation experiment, and 2) benchmark experiment as training set
	Y	1949	1949	
	T	1344	1344	

Positive Datasets: Phosphorylation sites of a specific window size are extracted, resulting in a positive dataset for each type of sites (S, T, and Y). Since independent observations are assumed by the CRUM learning algorithms, each positive training dataset is reduced based on homology. As done in PredPhospho [20], each

dataset is individually searched for sites with more than 70% identity using Needleman-Wunsch alignment [26], and only one of the similar sites is kept in the dataset. Homology reduction is not performed on the benchmark test dataset so that all sites are used in the evaluation, including sites at the edges of the protein sequence that do not have sufficient window size.

Negative Datasets: All S, T, and Y sites not annotated as phosphorylatable in the datasets' proteins are assumed to be non-phosphorylatable. This assumption may prove false in some instances, as some of the sites may indeed be undiscovered phosphorylation sites. However, due to the rareness of phosphorylation sites, only a few false negative sites are expected and will only affect our training and evaluation to a small degree [2]. Using the same procedure as the positive training datasets preparation, the negative training datasets are homology reduced. In addition, since the sizes of the negative datasets far exceed those of the positive datasets, only a random sample of the negative sites are chosen in equal size to each positive dataset to improve training [2, 20].

4.2 Performance Assessment Measures

The following predictive performance measures on the test datasets are used to assess all predictors: sensitivity (*Sn*), specificity (*Sp*), Matthews correlation coefficient (*MCC*), and accuracy (*Ac*). By construction, the test datasets are independent of the training datasets and so the test datasets have no bearing on the training or on the determination of the parameters. Thus, the test datasets gives unbiased performance measurements on the generalization ability of the considered predictors. The performance measures are computed as follows

$$Ac = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \tag{22}$$

$$Sn = \frac{TP}{TP + FN} \times 100 \tag{23}$$

$$Sp = \frac{TN}{TN + FP} \times 100 \tag{24}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \tag{25}$$

where the variables are the number of true positives (*TP*), false positives (*FP*), true negatives (*TN*), and false negatives (*FN*). For *Ac*, *Sn*, and *Sp*, the ideal is 100%. As a correlation coefficient, *MCC* ranges from –1 to +1, and the ideal predictor would have a *MCC* of 1.

4.3 Implementation

To conduct the computational experiments, we use the LIBSVM [8] implementation of the SVM that is freely available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. For the RVM, Tipping's MATLAB implementation of the fast RVM learning algorithm described in [38] is used, and is freely available at www.vectoranomaly.com. Since the RVM implementation is in MATLAB and a MATLAB interface is available for LIBSVM, we implemented CRUM, and conducted all dataset processing and experiments in the MATLAB environment.

5. RESULTS AND DISCUSSION

5.1 Comparison of Learning Methods

This experiment focuses on the comparison of learning methods where the performance of the CRUM learning method is compared to other popular kernel machines – the SVM and the RVM [37] – for prediction of phosphorylation sites. In addition, three different model selection schemes for the CRUM are compared.

5.1.1 Parameter Search

All models considered contain parameters that are not learned and thus must be determined by external means. These parameters are C and γ for the SVM, γ for the RVM, and M for the CRUM. A search over the parameter space is conducted and each model is evaluated using cross-validation. The parameters that result in the lowest prediction error according to cross-validation are used to train the final predictor for the comparison. For the T and Y datasets, 10-fold cross-validation is used. Since the S dataset is significantly larger, 3-fold cross-validation is used.

The parameter searches are conducted as follows. For the SVMs, we conduct a course grid search over $C = 2^i$ and $\gamma = 2^j$ where $i, j \in \{-10, -9, \ldots, 9, 10\}$. Based on the cross-validation results, a region is heuristically determined for a more precise parameter search. The parameter searches for RVM and CRUM are conducted in a similar fashion over a single dimension, γ or M, respectively. Also, the use of clustering in CRUM as described in Section 3.3 allows M to be selected as the model with the best AIC or BIC value.

5.1.2 Protein Site Features

Crystallization studies show that a region of the protein substrate of about 9 to 12 amino acids in length surrounding the acceptor residue contacts the PK active site [34]. Therefore, the sequence of amino acids surrounding a site in its 3D structure should provide sufficient information to determine whether the site is phosphorylatable. However, since the 3D structure is not usually known, most predictors settle for only using the amino acid sequence within a window surrounding the site in the linear sequence.

Since CRUM relies mainly on sequence like NetPhos, we adopt the same window sizes that they experimentally verified to have good predictive ability [3]. Specifically, a window size of 9 is used for T and Y sites, while a window size of 11 is used for S sites. Site sequences are represented using the BLOSUM encoding that was shown to achieve better performance than the orthogonal encoding [19]. Each amino acid in the window surrounding the site is represented by a real-valued 20-dimensional vector of substitution scores for each amino acid. A min-max normalized BLOSUM62 matrix is used, where each row of the matrix is the substitution scores of the amino acid against all other amino acids. Using a window size of 11 for S sites, we need to represent each of the 11 amino acids downstream and upstream of the site, resulting in a 440-dimensional vector representing the site. Similarly, T and Y sites are represented by 360-dimensional vectors. Note that center amino acid is always an S, T, or Y, and is thus excluded from the representation.

5.1.3 Predictor Assessment

In addition to the performance assessment measures described in Section 4.2, the predictors are also assessed based on their computational complexity. The number of critical vectors (CVs), which are the SVs, RVs, and RUs for the SVM, RVM, and CRUM, respectively, are reported, as this measure gives an indication of the run-time and memory complexity of the resulting predictors, as given in Equations (2), (3), and (4). We also conduct run-time benchmarking trials for the training and prediction algorithms, and report the average of 10 trials. In this case, note that there is a bias toward the SVM implemented in faster compiled C++, whereas both CRUM and RVM are implemented in the slower interpreted language of MATLAB. Due to the presence of multiple local minima in the clustering phase, the CRUM predictors includes 10 replicates of k-means to mitigate this problem and is included in the training run-times reported.

The computer used on the smaller T and Y datasets is a 2.53 GHz Intel Core 2 Duo with 4 GB of memory. Due to the memory requirements for the RVM, we used a PC with 2.83 GHz Intel Core 2 Quad with 8 GB of memory to analyze the large S dataset. The run-time benchmarking trials are run on these computers.

5.1.4 Results

For this experiment, we used the evaluation dataset described in Section 4.1. For an unbiased comparison between different models, we need a test dataset. We therefore reserve 10% of each of the positive and negative datasets as the test dataset. The remaining 90% forms the training and validation datasets.

Table 2, Table 3, and Table 4 show the details of the comparative analysis of the predictors on the test dataset for S, T, and Y site predictions, respectively. The CRUM predictor names indicate the method used to determine the model complexity parameter *M*, which is either "CV" for cross-validation, AIC, or BIC.

In Table 2, it is shown that the SVM has the best predictive performance for the S dataset, but the CRUM and RVM are close. The SVM also has the shortest training times, but we note again that the reported times are biased toward the SVM due to its faster compiled implementation compared to the slower interpreted language of MATLAB. However the CRUM and RVM models are significantly more parsimonious with the SVM having at least 67 times more CVs. The effect this complexity has can be seen on the testing times where the RVM and CRUM predictors are at least 9 times faster even with the implementation disadvantage.

With Table 3 for the T dataset, it is shown again that the SVM achieves the highest predictive performance. However, the RVM and CRUM predictors perform closely. The CRUM and RVM models are again more parsimonious than the SVM, using between 0.66% and 2.90% of the number of CVs. This leads to CRUM and RVM predictors that are 9 to 31 times faster than the SVM. Furthermore the CRUM training time is about 4 to 7 times faster than the RVM training.

Table 4 shows that the CRUM predictors have better predictive performance over the SVM and RVM, though all three methods are close. We again see the CRUM and RVM models achieving more parsimonious models than the SVM model, using between 0.68% to 3.33% of the number of CVs; leading to prediction run-time on the testing data of about 8 to 41 times faster than on the SVM. In this case, the RVM has a fast training time, beating the CRUM-AIC predictor by a few seconds. However, the CRUM-BIC predictor's, with comparable predictive performance and complexity, training is about 2 times faster than RVM's and only about 3 times slower than SVM's.

Table 2. Comparison of predictors on test dataset for phosphoserine (S) prediction.

Predictor	Ac (%)	Sn (%)	Sp (%)	MCC	#CV	Train (s)	Test (s)
SVM	75.32	68.42	81.54	0.51	10536	491.42	32.46
RVM	74.01	69.52	78.08	0.48	76	3312.55	2.03
CRUM-CV	73.97	69.63	77.89	0.48	134	2964.50	2.27
CRUM-BIC	70.34	63.82	76.21	0.40	64	2681.57	1.10
CRUM-AIC	74.13	68.42	79.27	0.48	157	3135.63	2.74

Table 3. Comparison of predictors on test dataset for phosphothreonine (T) prediction.

Predictor	Ac (%)	Sn (%)	Sp (%)	MCC	#CV	Train (s)	Test(s)
SVM	76.55	60.45	89.25	0.53	3031	25.80	2.16
RVM	73.75	60.90	83.87	0.46	50	499.68	0.19
CRUM-CV	74.35	59.09	86.38	0.48	82	129.66	0.25
CRUM-BIC	69.74	53.64	82.44	0.38	20	73.80	0.07
CRUM-AIC	73.74	57.73	86.38	0.47	88	139.56	0.25

Table 4. Comparison of predictors on test dataset for phosphotyrosine (Y) prediction.

Predictor	Ac (%)	Sn (%)	Sp (%)	MCC	#CV	Train (s)	Test(s)
SVM	68.52	64.52	72.19	0.37	2339	11.10	1.24
RVM	69.75	65.16	73.96	0.39	16	75.95	0.04
CRUM-CV	70.06	65.81	73.96	0.40	37	53.07	0.07
CRUM-BIC	70.37	66.45	73.96	0.41	16	36.91	0.03
CRUM-AIC	70.37	66.45	73.96	0.41	78	78.31	0.17

Overall, we see that the SVM offers the best prediction accuracy of the methods considered for S and T prediction and the quickest training times over all datasets. However, this comes at the significant cost that the user does not know the uncertainty of the SVM's prediction, or has an inaccurate measure of uncertainty if Platt's method is used to obtain posterior estimates from SVM outputs [28, 37]. Both the CRUM and RVM provide accurate posterior estimates by virtue of their data likelihood-based objective functions [25, 37], while also significantly reducing the run-time and memory complexity of the prediction algorithm, as indicated by the use of less than 5% of the CVs than the SVM. Arguably, prediction run-time that is more important than training run-time, as training is typically invoked at far less frequency than the prediction algorithm.

In comparison to the RVM, the CRUM achieves comparable predictive performance with a lower cost training algorithm, in both run-time and memory complexity. Furthermore, through the use of AIC and BIC, model selection is simpler and faster with the CRUM. It is known that with a finite sample, BIC underestimates the model complexity, while the AIC overestimates it [16], as confirmed here by the cross-validation model complexity falling between BIC's and AIC's. However the results also show that this is not an issue with the CRUM, as the built-in regularization mitigates the overfitting problem and the resulting complex AIC model exhibits comparable results to the cross-validation model. Therefore the use of AIC is a cheap, viable option for CRUM model selection.

5.2 Evaluation of Protein Site Features

The goal of this experiment is to evaluate different phosphorylation site features and their combinations. In addition to the BLOSUM encoded sequence feature described in Section 5.1.2, we consider amino acid composition and predicted disorder regions, which provide 3D structure information. The performance of CRUM predictors based on combinations of these three features is compared via cross-validation to give insight into which feature combinations are important.

5.2.1 Protein Site Features

BLOSUM Encoded Sequence Feature: See Section 5.1.2.

Amino Acid Composition: A statistical analysis of the amino acids surrounding phosphorylation sites revealed characteristics that coincide with the amino acid composition of disordered regions. Both types of sequences are significantly depleted in rigid amino acids and significantly enriched with flexible amino acids [18]. Due to this statistically significant difference in composition, amino acid composition can be used to discriminate a phosphorylatable site from a non-phosphorylatable site. The percent amino acid composition of the site can be represented using a 20-dimensional vector, where each component represents the percent composition of one of the 20 amino acids. The percent composition is computed over the same window size as the extracted site, including the center amino acid.

Disordered Regions: Phosphorylation sites and other post-translational modifications are observed to be located in disordered regions of proteins frequently, but not exclusively [18]. Intrinsically disordered protein regions are the parts of the protein that do not fold into a stable 3D structure. To maximize the applicability of CRUM, the true identification of disordered regions is not required and is instead predicted using a disorder predictor. The SVM and logistic regression-based VSL2B predictor is used that is based on amino acid composition [30]. VSL2B is applied to the entire protein sequence using an output window length of one, resulting in a probabilistic classification for each amino acid in the sequence being in a disorder region. The disorder classifications of the site and its surrounding amino acids are extracted over the same window sizes as the extracted site sequence. Therefore, 23-dimensional vectors represent S sites, and 19-dimensional vectors represent T and Y sites. Unlike the BLOSUM encoding, the center amino acid is included, as its disorder classification is dependent on the surrounding amino acids.

5.2.2 Results

To evaluate the predictive performance of using the CRUM under a variety of different protein site features, we conducted a 3-fold cross-validation experiment using a benchmark training dataset described in Section 4.1. The protein features considered are the BLOSUM encoding of the site (B), disorder (D), amino acid composition (A), and combinations thereof. Model selection for these CRUM predictors is conducted on the entire benchmark training dataset using the AIC scheme. Table 5, Table 6, and Table 7 give the resulting performance for S, T, and Y sites respectively, using a posterior threshold of 0.5. This means we classify a site as phosphorylatable if the predicted posterior probability is greater than 0.5, otherwise the site is classified as not phosphorylatable.

The results indicate that the CRUM predictors containing the BLOSUM feature show the best performance in terms of both accuracy and Matthews correlation coefficient (MCC) with well-balanced sensitivity and specificity. Excluding the BLOSUM feature drops the specificity, and thus overall accuracy and MCC, by including more false positives. This remains true under a variety of different posterior threshold settings, as plotted in Figure 3 through Figure 5. In the ROC plots, the four CRUM predictors that use the BLOSUM feature (CRUM B, B+A, B+D, and B+D+A) perform roughly identically under different threshold settings, and are always better or equal to the predictors that exclude the BLOSUM feature. This experiment suggests that using the BLOSUM feature is important to achieve good predictive performance.

Table 5. Three-fold cross-validation performance on phosphoserine (S) prediction. (Ac – Accuracy, Sn – Sensitivity, Sp – Specificity, MCC – Matthews correlation coefficient; B – BLOSUM encoding feature, D – Disorder feature, A – Amino acid composition feature).

Features	Ac (%)	Sn (%)	Sp (%)	MCC
CRUM B	75.25	72.33	78.18	0.51
CRUM D	64.00	78.36	49.64	0.29
CRUM A	69.08	70.04	68.12	0.38
CRUM B + D	75.49	73.16	77.82	0.51
CRUM B + A	75.21	72.35	78.07	0.51
CRUM D + A	68.37	72.73	64.00	0.37
CRUM B + D + A	75.60	73.32	77.88	0.51

Table 6. Three-fold cross-validation performance on phosphothreonine (T) prediction. (Ac – Accuracy, Sn – Sensitivity, Sp – Specificity, MCC – Matthews correlation coefficient; B – BLOSUM encoding feature, D – Disorder feature, A – Amino acid composition feature).

Features	Ac (%)	Sn (%)	Sp (%)	MCC
CRUM B	71.34	66.24	76.45	0.43
CRUM D	64.91	71.88	57.93	0.30
CRUM A	67.75	66.44	69.06	0.36
CRUM B + D	71.73	67.21	76.24	0.44
CRUM B + A	71.19	66.09	76.30	0.43
CRUM D + A	68.06	69.06	67.06	0.36
CRUM B + D + A	71.83	67.21	76.46	0.44

Table 7. Three-fold cross-validation performance on phosphotyrosine (Y) prediction. (Ac – Accuracy, Sn – Sensitivity, Sp – Specificity, MCC – Matthews correlation coefficient; B – BLOSUM encoding feature, D – Disorder feature, A – Amino acid composition feature).

Features	Ac (%)	Sn (%)	Sp (%)	MCC
CRUM B	66.56	66.07	67.04	0.33
CRUM D	62.31	66.89	57.74	0.25
CRUM A	62.65	63.91	61.38	0.25
CRUM B + D	65.74	68.15	63.32	0.32
CRUM B + A	66.70	67.11	66.29	0.33
CRUM D + A	62.69	67.26	58.11	0.25
CRUM B + D + A	66.00	67.63	64.36	0.32

It also should be noted that a user can target a desired false positive rate by appropriately selecting the posterior threshold for the CRUM using the ROC plots. No retraining is needed. This is in contrast with predictors like PPRED [2] that uses the standard hard-decision SVMs with no additional trained logistic regression model on the SVM outputs [28], where the specificity is fixed after training and thus the user has no way to change it without retraining the predictor under different settings of parameters, such as window size or positive to negative training data ratio.

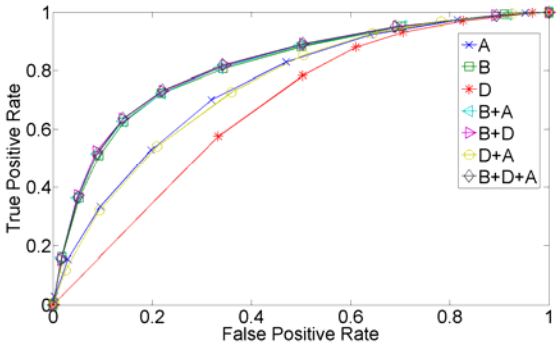

Figure 3. ROC for phosphoserine (S) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

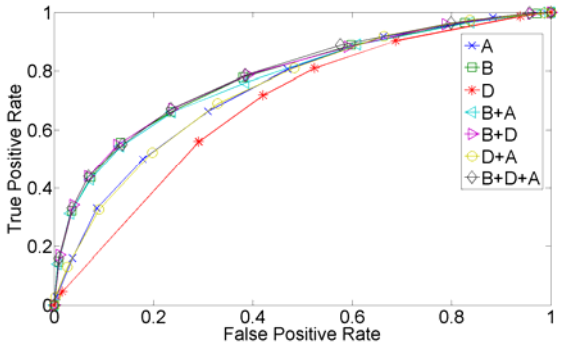

Figure 4. ROC for phosphothreonine (T) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

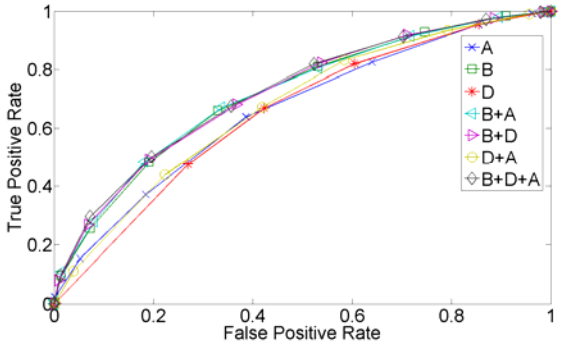

Figure 5. ROC for phosphotyrosine (Y) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

5.3 Benchmark Comparison

Finally, the third experiment concentrates on the comparative study with existing predictors where the performance of different CRUM predictor configurations is compared to that of existing predictors on previously published benchmark datasets [2]. For this experiment, the CRUM predictors using different combinations of the features described in Section 5.2.1 are trained on the entire benchmark training dataset described in Section 4.1. The trained CRUM predictors' performances are then computed using the benchmark test dataset that is independent of the training set, as described in Section 4.1.

For comparison, the performance of NETPHOS [3] and DISPHOS [18] are also computed using the latest versions available on the web. Note that these tools have been updated since their original publication and may have been trained on proteins included in the benchmark test dataset used in this experiment. Therefore, the performance of NETPHOS and DISPHOS may be biased in favor of them in this comparison.

5.3.1 Protein Features for Sites on Edges of Protein

In previous experiments and in the training of the CRUM predictors in this section, sites that lie on the edges of a protein sequence are omitted because of the insufficient number of amino acids to fill the required window size. However, since NETPHOS

and DIPHOS report results for these edges sites, we modify CRUM to do the same when computing predictions.

Edge sites are accommodated by filling in the missing amino acids with the ambiguous amino acid code X. If the site is found near the start of a protein sequence, then the appropriate number of Xs are prepended to the site's sequence to fill the window size. If the site is found near the end of the protein sequence, then the appropriate number of Xs are appended to the site's sequence to fill the window size. These Xs are ignored in the computation of amino acid composition and are given a disorder value of 0.5 for its position in the disorder encoding described in Section 5.2.1.

5.3.2 Results

Figures 6, 7, and 8 show ROC plots for S, T, and Y sites, respectively. Similar to the cross-validation experiment, we see that all the BLOSUM-based CRUM predictors perform comparably with generally greater TPRs compared to the non-BLOSUM-based CRUM predictors (A, D, and D+A) at similar FPR values.

In comparison with NETPHOS and DISPHOS, there is no uniformly better tool in this benchmark test dataset. NETPHOS underperforms DISPHOS and the four BLOSUM-based CRUM predictors uniformly over all FPR values.

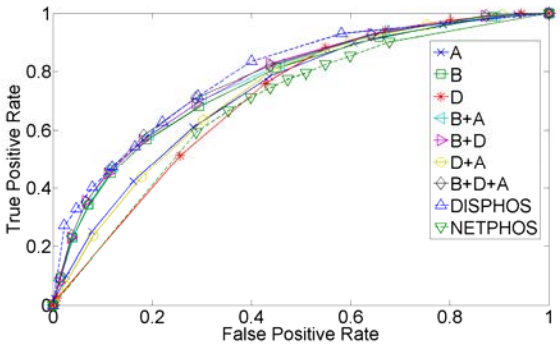

Figure 6. ROC for phosphoserine (S) prediction based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

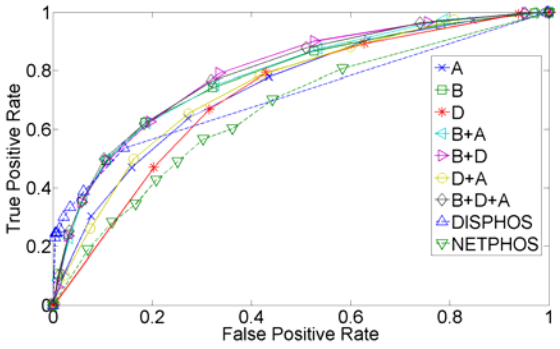

Figure 7. ROC for phosphothreonine (T) prediction based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

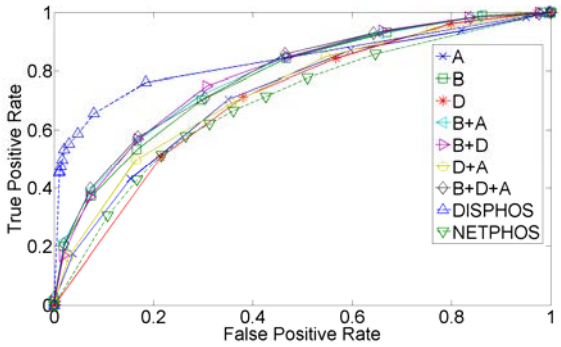

Figure 8. ROC for phosphotyrosine (Y) prediction based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

For S sites in Figure 6, DISPHOS has an advantage in the very small FPR range (less than ~0.03) and at around 0.4 - 0.6 FPR, but otherwise the performance is comparable to the simpler BLOSUM-based CRUM predictors. For T sites in Figure 7, we again see DISPHOS and BLOSUM-based CRUM predictors being comparable, with the exception of an advantage for DISPHOS at the very low FPR range, again at less than ~0.03. However, different characteristics are seen for the Y sites in Figure 8. Here DISPHOS displays the best performance for FPRs under 0.6, with an increase in TPR of at most ~0.3 compared to BLOSUM-based CRUM predictors.

A possible major factor for the underperformance of the CRUM predictors is the high dimensionality of the site representation relative to the sample size, particularly on the Y dataset. The design of DISPHOS' encoding of sites into vectors is more computationally complex, such as the inclusion of the results of several disorder predictors, compared to the settings used here from CRUM. However, the design of DISPHOS used Fisher's permutation test and principal component analysis to reduce the dimensionality [18]. This dimensionality reduction to account for the small sample size may be a major factor in DISPHOS' performance on the Y dataset, and will be considered in future work using CRUM.

6. CONCLUSIONS

In this study, probabilistic predictions of protein phosphorylation sites were proposed using our CRUM model as an alternative to the popular SVM and RVM. The experimental results show that the extremely compact models of CRUM provide a significant reduction to the computational complexity of the prediction algorithm, which is advantageous in conducting predictions at a large scale. We show that CRUM using only the BLOSUM feature, CRUM B, has among the best accuracy and MCC performance compared to other CRUM settings in both cross-validation and benchmark experiments. The simplicity and efficiency of the BLOSUM feature computation make the data preparation process for using CRUM faster, as disorder predictors do not need to be invoked.

While the BLOSUM-based CRUM predictors are often comparable to the more complex DISPHOS predictor, which uses several disorder and other structural predictors to extract features, on the S and T benchmark datasets, there is a cost to this

reduction of computational complexity, as DISPHOS outperforms the different CRUM predictor configurations considered in this study under the Y benchmark dataset.

In the future, as mentioned previously, dimensionality reduction of the protein feature encoding will be considered to improve the performance, particularly with Y sites. In addition, a multi-kernel implementation of CRUM shall be considered where only a single protein feature will be considered in each kernel. Currently in the combined feature CRUM predictors (e.g. B+D+A), all features are concatenated into the one vector and used in the same Gaussian kernel. When combined, it is possible that the lower dimensional features D and A may be overshadowed by the B feature simply due to B's high dimensionality rather than D or A being unimportant, as the above results may suggest. A multi-kernel solution may avoid this issue and the supervised learning algorithm may then be able to exploit all features for improved predictive performance.

Furthermore, we plan to extend the CRUM model and training algorithm for multi-class prediction. This will allow for kinase-specific phosphorylation site prediction, where a prediction for what type of PK phosphorylates the site is also made.

7. ACKNOWLEDGMENTS

This work is supported in part by NIH Grants from the National Institute of General Medical Sciences (P20GM103516 and P20GM103466). The paper's contents are solely the responsibility of the authors and do not necessarily represent the official views of the NIH.

8. REFERENCES

[1] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Biswas, A. K., Noman, N., and Sikder, A. R. Machine learning approach to predict protein phosphorylation sites by incorporating evolutionary information. *BMC Bioinformatics* 2010, **11**:273.

[3] Blom, N., Gammeltoft, S., and Brunak, S. Sequence and structure-based prediction of eukaryotic protein phosphorylation sites. *Journal of Molecular Biology* 1999, **294**(5):1351-1362.

[4] Blom, N., Sicheritz-Pontén, T., Gupta, R., Gammeltoft, S, and Brunak, S. Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence. *Proteomics* 2004, **4**(6):1633-1649.

[5] Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M. C., Estreicher, A., et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research* 2003, **31**(1):365-370.

[6] Boser, B. E., Guyon, I. M., and Vapnik, V. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on computational learning theory (COLT)*. ACM Press, 1992, 144-152.

[7] Brinkworth, R. I., Breinl, R. A., and Kobe, B. Structural basis and prediction of substrate specificity in protein serine/threonine kinases. *PNAS* 2003, **100**(1):74-79.

[8] Chang, C.-C. and Lin, C.-J. LIBSVM: a library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* 2011, **2**(3):27.

[9] Corduneanu, A., and Bishop, C. M. Variational Bayesian model selection for mixture distributions. In Jaakkola, T., Richardson, T., editors, *Artificial Intelligence and Statistics*, Morgan Kaufmann, 2001, 27-34.

[10] Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning* 1995, **20**(3):273-297.

[11] Dinkel, H., Chica, C., Via, A., Gould, C. M., Jensen, L. J., Gibson, T. J., and Diella, F. Phospho.elm: a database of phosphorylation sites – update 2011. *Nucleic Acids Research* 2011, **39**:D261-D267.

[12] Diella, F., Gould, C. M., Chica, C., Via, A., and Gibson, T. J. Phospho.elm: a database of phosphorylation sites – update 2008. *Nucleic Acids Research* 2008, **36**:D240-D244.

[13] Eck, M. J. and Manley, P. W. The interplay of structural information and functional studies in kinase drug design: insights from BCR-Abl. *Current Opinion in Cell Biology* 2009, **21**(2):288-295.

[14] Gao, J., Thelen, J. J., Dunker, A. K., and Xu, D. Musite, a tool for global prediction of general and kinase-specific phosphorylation sites. *Molecular & Cellular Proteomics* 2010, **9**(12):2586-2600.

[15] Gao, J. and Zhang, J. Sparse kernel learning and the relevance units machine. In *Proceedings of the 13th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD '09)*. Springer-Verlag, 2009, 612-619.

[16] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[17] Hornbeck, P. V., Chabra, I., Kornhauser, J. M., Skrzypek, E., and Zhang, B. PhosphoSite: A bioinformatics resource dedicated to physiological protein phosphorylation. *Proteomics* 2004, **4**(6):1551-1561.

[18] Iakoucheva, L. M., Radiojac, P., Brown, C. J., O'Connor, T. R., Sikes, J. G., Obradovic, Z., and Dunker, A. K. The importance of intrinsic disorder for protein phosphorylation. *Nucleic Acids Research* 2004, **32**(3):1037-1049.

[19] Ingrell, C. R., Miller, M. L., Jensen, O. N., and Blom, N. Netphosyeast: prediction of protein phosphorylation sites in yeast. *Bioinformatics* 2007, **23**(7):895-897.

[20] Kim, J. H., Lee, J., Oh, B., Kim, K., and Koh, I. Prediction of phosphorylation sites using svms. *Bioinformatics* 2004, **20**(17):3179-3184.

[21] Kobe, B., Kampmann, T., Forwood, J. K., Listwan, P., and Brinkworth, R. I. Substrate specificity of protein kinases and computational prediction of substrates. *Biochimica et Biphysica Acta* 2005, **1754**(1-2):200-209.

[22] Koltchinskii, V. Rademacher penalties and structural risk minimization. *IEEE Transactions of Information Theory* 2001, **47**:1902-1914.

[23] Kreegipuu, A., Blom, N., and Brunak, S. PhosphoBase, a database of phosphorylation sites: release 2.0. *Nucleic Acids Research* 1999, **27**(1):237-239.

[24] MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*. University of California Press, 1967, 281-297.

[25] Menor, M. and Baek, K. Relevence units machine for classification. In *Proceedings of the 4th international conference on BioMedical Engineering and Informatics (BMEI '11)*. IEEE, 2011, 2295-2299.

[26] Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 1970, **48**(3):443-453.

[27] Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, 2006.

[28] Platt, J. C. Probabilities for SV machines. In Smola, A. J., Bartlett, P. L., Scholkopf, B., et al., editors, *Advances in Large Margin Classifiers*, MIT Press, 2000, 61-73.

[29] Pelleg, D. and Moore, A. X-means: extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th international conference on machine learning*. Morgan Kaufmann, 2000, 727-734.

[30] Peng, K., Radivojac, P., Vucetic, S., Dunker, A. K., and Obradovic, Z. Length-dependent prediction of protein intrinsic disorder. *BMC Bioinformatics* 2006, **7**:208.

[31] Rosset, S., Zhu, J., and Hastie, T. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems 15 (NIPS '03)*. MIT Press, 2003.

[32] Shawe-Taylor, J. and Cristianini, N. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[33] Sikder, A. R. and Zomaya, A. Y. Analysis of protein phosphorylation site predictors with an independent dataset. *International Journal of Bioinformatics Research and Applications* 2009, **5**(1):20-37.

[34] Songyang, Z., Blechner, S., Hoagland, N., Hoekstra, M. F., Piwnica-Worms, H., and Cantley, L. C. Use of an oriented peptide library to determine the optimal substrates of protein kinases. *Current Biology* 1994, **4**(11):973-982.

[35] Still, S. and Bialek, W. How many clusters? An information-theoretic perspective. *Neural Computing* 2004, **16**:2483-2506.

[36] Tang, Y. R., Chen, Y. Z., Canchaya, C. A., and Zhang, Z. GANNPhos: a new phosphorylation site predictor based on a genetic algorithm integrated neural network. *Protein Engineering, Design & Selection* 2007, **20**(8):405-412.

[37] Tipping, M. E. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 2001, **1**:211-244.

[38] Tipping, M. E. and Faul, A. C. Fast marginal likelihood maximization for sparse Bayesian models. In Bishop, C. M., Frey, B. J., editors, *Proceedings of the 9th international workshop on artificial intelligence and statistics*, 2003.

[39] Trost, B. and Kusalik, A. Computational prediction of eukaryotic phosphorylation sites. *Bioinformatics* 2011, **27**(21):2927-2935.

[40] Vapnik, V. and Lerner, A. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 1963, **24**:744-780

[41] Wong, Y. H., Lee, T. Y., Liang, H. K., Huang, C. M., Wang, T. Y., et al. KinasePhos 2.0: a web server for identifying protein kinase-specific phosphorylation sites based on sequences and coupling patterns. *Nucleic Acids Research* 2007, **35**:W588-W594.

[42] Xue, Y., Gao, X., Cao, J., Liu, Z., Jin, C., Wen, L., Yao, X., and Ren, J. A summary of computational resources of protein phosphorylation. *Current Protein and Peptide Science* 2010, **11**(6):485-496.

ABOUT THE AUTHORS:

Mark Menor is currently a doctoral student in the Department of Information and Computer Sciences at the University of Hawai`i at Mānoa. He received his B.S. and M.S. degrees in Computer Science from the University of Hawai`i at Mānoa in 2005 and 2007. His current research interests include bioinformatics and machine learning.

Kyungim Baek received the B.S. degree in Mathematics and the M.S. degree in Computer Science from Sogang University, Seoul, Korea, in 1994 and 1996, and the Ph.D. degree in Computer Science from Colorado State University, Fort Collins, in 2002. During 2002-2005, she was a Postdoctoral Research Fellow in the Laboratory for Intelligent Imaging and Neural Computing (LIINC) in the Department of Biomedical Engineering at Columbia University, New York. She is currently an assistant professor of the Department of Information and Computer Sciences at the University of Hawai`i at Mānoa. Her research interests include pattern recognition and machine learning applied to bioinformatics, computational models of visual perception, neural computation, and related applications in computer vision.

Guylaine Poisson received a B.S. degree and a M.S. degree in Biological Sciences from the Univeristé de Montréal, Québec, Canada in 1994 and 1997, and a Ph.D. degree in Cognitive Computer Science from the Université du Québec à Montréal, Québec, Canada in 2005. She is currently an associate professor in the Department of Information and Computer Sciences at the University of Hawai`i at Mānoa. Her research interests include bioinformatics and computational biology.