Classification of Non-Ribosomal Peptides Using Support Vector Machines

Hannah Acheson-Field, Eric Chu

December 2013

Contents

1 Introduction

Non-ribosomal peptides (NRP's) are proteins synthesized in a non-ribosomal process. This process generally occurs in bacterial and fungi organisms. These peptides are composed of an enzyme and a number of modules. One of these modules is an A-domain, which is responsible for determining which amino acids will make up the final product. Because A-domains are responsible for determining the final product, the sequence of A-domains will represent the sequence of amino acids that is produced by the Non-ribosomal peptide.

The data set used in this project is composed of 1547 A-domain sequences and the substrate its corresponding protein binds to. Each sequence is about 400 amino acids long. We use support vector machines to create a classifier that maps a given A-domain to a substrate product. The data set is the same dataset used by Prieto et al. and Sondergaard et al. (2) (5)

A support vector machine (SVM) is a machine learning classification algorithm based on finding the optimal hyperplane between two sets of data points. The kernel method is a technique employed to transform non-linearly separable data into seperable data by mapping the data into a higher dimension.

Through evaluation of different feature sets, our approach should shed further light on which properties of A-domains are particularly important in determining protein function. This relationship is one piece in proteomics that is open for investigation. Not unlike attempts at de novo protein structure prediction, a classifier, albeit an imperfect one, can serve as a time-saving tool that limits the space in which researchers may have to search through. Ultimately, greater understanding about sequences, A-domains, and protein substrates can

aid in both creating drugs that target specific proteins and designing novel proteins. Being an extension of previous approaches, this project also adds to the existing corpus of the efficacy of different machine learning approaches.

This report includes the theory behind support vector machines, different approaches to representing our features, and the accuracy of our predictions using both k-fold cross-validation and stratified k-fold cross-validation.

1.1 Previous Approaches

Previous approaches to classify the NRP's include the use of Position Specific Scoring Matrices (PSSM's), Hidden Markov Models (HMM's), and Neural Networks (NN's). (2) (5)

1.1.1 Hidden Markov Models

Hidden Markov Models are statistical models that are keyed by the assumption of the Markov property, which dictates that future states are only dependent on the present state and not on the past. This is commonly referred to as a 'memoryless' system. 'Hidden' refers to the fact that the states are not directly observable themselves but must be inferred from their outputs. States have transition parameters and output parameters controlling the likelihood of the next state and the likelihood of a particular output.

Previous approaches to our problem using HMM's relied on comparing the sequences against HMM profiles. In this case, the HMM is composed of match, insertion, and deletion states, reflecting how a sequence may evolve into another sequence. The model's transitions form a left-to-right structure with no cycles. The goal is to find transition probabilities that maximize the likelihood of transitions between sequences that bind to the same substrate. Note that the number of match states is another parameter to determine, as the possibility of insertions and deletions may change the length of the sequence. A HMM profile is created for each substrate family.

Next, each dataset sequence can be tested against each HMM profile. Different techniques exist for scoring a profile. Due to the computational complexity often required in calculating the probability of acquiring the given sequence from the match states sequence, the most common method is a dynamic programming algorithm called the forward algorithm. A sequence may then be matched to the highest scoring profile, or perhaps not at all if none pass a desired threshold. More detailed information about HMM's and HMM profiles in biological contexts can be found in Yoon's survey paper. (15)

1.1.2 Neural Networks

Instead of HMM's, Sondergaard et. al used neural networks, which were originally inspired by brain neurons. A neural network is composed of neurons that process a set of inputs through some weighted combination, pass the result to an activation function, and output that result to the next set of neurons. The

activation function most commonly used is the sigmoidal function. The network itself is composed of a input layer of neurons, any number of hidden layers, and an output layer. The goal is to, whether through a method such as back propagation or gradient descent, determine an optimal set of weights for each neuron. A trained neural network thus takes an input vector $\mathbf{x}$ and outputs a vector $\mathbf{y}$.

An examination of Sondergaard et. al's approach may provide some clarification. They encoded their input by first converting each sequence to a vector counting the occurence of each possible n-gram. The network's input layer contains 20^n neurons, corresponding to the number of possible n-grams (the length of the encoded sequence vector). The outputs of the input layer neurons are passed according to the network topology to the first of three hidden layers, each of which contains 10 neurons. Finally, the outputs of the third hidden layer are passed to the output layer, which consists of 30 neurons. Each of these 30 neurons corresponds to one of the substrate families. Thus, their values can be evaluated as a posterior probability of that substrate given the input sequence. Naturally, the sequence is classified as the substrate family with the highest score.

2 Support Vector Machine Theory

In this section, we will provide theory behind SVM's. We will discuss binary classification of fully seperable data, binary classification of non-fully seperable data, SVM's used with regression, non-linear SVM's, and kernel functions. We will also discuss our rationale behind using SVM's for this problem. The following material is adapted from the Tristan Fletcher's *SVM's Explained* (6).

2.1 Binary Classification of Fully Seperable Data

We describe our data as follows: $\{\mathbf{x}_i, y_i\}$, when i=1...L, $y_i \in$ {-1, 1}, and $x_i \in R^D$. Because the data is linearly seperable, we can draw some line (when D=2) or some hyperplane (when D >2).

This hyperplane can be described as: $\mathbf{x} \cdot \mathbf{w}$ + b = 0, where $\mathbf{w}$ is normal to the hyperplane, and where $\frac{b}{\|\mathbf{w}\|}$ is the distance between the hyperplane and the origin. SVM's find $\mathbf{w}$ and b such that the two equations are satisfied:

$$\mathbf{x_i} \cdot \mathbf{w} + b \leq -1$$
$$\mathbf{x_i} \cdot \mathbf{w} + b \geq 1$$

These equations can be combined as follows:

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geq 0 \ \ \forall \ i$$

We now consider the points on either side of the hyperplane which are closest to the hyperplane. These points are the support vectors. We aim to maximize

this margin $\frac{1}{\|\mathbf{w}\|}$:

$$min_w \|\mathbf{w}\| \text{ subject to } y_i(\mathbf{x_i} \cdot \mathbf{w} + b) \geq 0 \ \ \forall \, i$$

This is the same as:

$$min_w \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y_i(\mathbf{x_i} \cdot \mathbf{w} + b) \geq 0 \ \ \forall \, i$$

We add a Lagrange multiplier α:

$$\begin{aligned} L_P &= \frac{1}{2}\|\mathbf{w}\|^2 - \alpha[y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \ \ \forall \, i] \\ &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \\ &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i(\mathbf{x_i} \cdot \mathbf{w} + b) + \sum_{i=1}^{N} \alpha_i \end{aligned}$$

We now try to find the $\mathbf{w}$ and b that minimize and the α which maximizes. We differentiate and set the derivatives to 0:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0, \ \mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i x_i$$

$$\frac{\partial L_P}{\partial b} = 0, \ \sum_{i=1}^{N} y_i \alpha_i = 0$$

Through substitution, we create a new equation. This is the dual form of the primary. It is dependent on α, and we maximize with respect to α.

$$\begin{aligned} L_D &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \text{ s.t. } \alpha_i \geq 0 \ \ \forall \, i, \ \sum_{i=1}^{N} y_i \alpha_i = 0 \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \mathbf{H_i}, \mathbf{j} \alpha_j, \text{ where } \mathbf{H_i}, \mathbf{j} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \alpha^T \mathbf{H} \alpha s.t. a_i \geq 0 \ \ \forall \alpha_i, \sum_{i=1}^{N} y_i \alpha_i = 0 \end{aligned}$$

Given the dual form of the primary, we can solve for α using quadratic optimization software. $\mathbf{w}$ can then be found using this α.

To find b, we consider the use of support vectors. Any support vector will have the form:

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) = 1$$

Substituting the equation for $\mathbf{w}$ into the equation above, we have:

$$y_i(\sum_{m \in s} \alpha_m y_m \mathbf{x_m} \cdot \mathbf{x_s} + b) = 1.$$

We then multiply through by y_s and use the fact that $y_s^2 = 1$ to obtain the following equation:

$$b = y_s - \sum_{m \in s} \alpha_m y_m \mathbf{x_m} \cdot \mathbf{x_s}$$

Generally, the average over all support vectors is used:

$$b = \frac{1}{N_S} \sum_{s \in S} (y_s - \sum_{m \in s} \alpha_m y_m \mathbf{x_m} \cdot \mathbf{x}_s)$$

We have now derived $\mathbf{w}$ and b for the optimal hyperplane.

2.2 Binary Classification of Non-Fully Seperable Data

To account for non-linearly separable data, we take into account misclassified data points. To do this, we introduce a slack variable ξ:

$$\mathbf{x_i} \cdot \mathbf{w} + b \leq -1 + \xi_i \text{ for } y_i = +1$$
$$\mathbf{x_i} \cdot \mathbf{w} + b \geq 1 - \xi_i \text{ for } y_i = -1$$

These equations can be combined to get:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi i \geq 0 \ \ \forall \, i.$$

We adapt our previous equation as follows:

$$min_w \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \text{ s.t. } y_i(\mathbf{x_i} \cdot \mathbf{w} + b) \geq 0 \ \ \forall \, i.$$

C is the variable determining the tradeoff between slack and penalty of margin size. Using this equation, the values for $\mathbf{w}$ and b can be calculated in similar fashion.

2.3 SVM's used with Regression

When using regression, our classifying values change to include all real numbers: $y_i \in R$. The penalty function will now be based on whether the difference between the predicted and actual value is less than a certain threshold ϵ: $\|t_i$ - $y_i\|$ $\leq \epsilon$. We introduce two additional penalties ξ^+ and ξ^- that indicate whether or not the value lies outside the ϵ-insensitive tube, the region bounded by $y_i \pm \epsilon \ \ \forall \, i$.

Our penalty functions are now:

$$t_i \leq y_i + \epsilon + \xi^+$$
$$t_i \geq y_i - \epsilon - \xi^-$$

Our objective funtion is now:

$$C\sum_{i=0}^{N}(\xi^{+}+\xi^{-})+\frac{1}{2}\|\mathbf{w}\|^{2}$$

, minimized subject to

$$\xi^{+}\geq 0 \text{ and } \xi^{-}\geq 0 \ \ \forall\, i$$

Values for **w** and b can again be calculated in similar fashion.

2.4 Non-Linear SVM's and Kernel Functions

Often times the dataset is not linearly separable. We can define $\mathbf{H}_{i,j}$ from earlier as:

$$\mathbf{H}_{i,j}=y_i y_j k(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}\cdot\mathbf{x_j}=\mathbf{x_i}^{T}\mathbf{x_j}.$$

k($\mathbf{x}_i$, $\mathbf{x}_j$) is an example of a kernel function. In this case, $\mathbf{x}_i^T\mathbf{x}_j$, is a linear kernel.

Kernel functions allow the feature space to be recast into higher dimensions. Oftentimes, once the feature space is recast, the data can be easily separated. Popular kernels include the polynomial kernel: $(\mathbf{x}_i^T\mathbf{x}_j+\mathrm{r})^d$, where r is the offset value and d is the degree, as well as the radial basis function (rbf) kernel: $e^{\frac{\|x_i-x_j\|^2}{2\sigma^2}}$, which will be discussed in greater detail later. Experimental results from our experimentation can be found in the results section.

2.5 VC Dimension

One reason SVM's first gained popularity was due to their theoretical groundings, which includes measurements of the Vapnik–Chervonenkis dimension. The VC dimension is a measure of the capacity of a learning algorithm and is defined as the greatest number of points that the algorithm can correctly classify, i.e. given any combination of class assignments to the points, the points will be able to separated by the SVM's hyperplane. For instance, the VC dimension of a line is three. In fact, any n-dimensional hyperplane has VC dimension of n+1. (12)

Once the VC dimension has been calculated, it can be used to give an upper bound on the error rate on test data. This is given by:

$$R(\alpha)\leq R_{emp}(\alpha)+\sqrt{\frac{h(log(\frac{2l}{h})+1)-log(\frac{\eta}{4})}{l}}$$

where *R_emp* is the error on the training data and η is the VC dimension. (1)

2.6 Other Topics Considered

We encounted a number of other techniques that could prove useful in augmenting our approach. Feature selection and dimensionality reduction are two sets of techniques that can produce faster runtimes, less noise, and better performance. Several methods exist. Recursive feature elimination, for example, is a greedy method in which the least useful feature is repeatedly removed. L1-based feature selection reduces the dimensionality by choosing features corresponding to non-zero alpha's, i.e. the points that are support vectors. The industry standard for feature selection is RELIEF, which runs in linear time in the number of given features and training instances, is applicable for binary or continuous data, and is relatively simple to understand and implement. Principal Component Analysis (PCA) is a popular technique to reduce dimensionality by finding an optimal hyperplane upon which to project data. Given the large dimensionality of some of our feature sets and the current ineffectiveness of combining those feature sets, these methods may merit a closer examination.

Another topic is kernel engineering, and more specifically, kernels specifically designed for computational biology. Recall that a kernel is simply a function that makes some implicit comparison between vecotrs; these kernels incorporate domain problem information into the comparison. As an example, the string matching kernel is designed to take into account that sequences are more similar the more subsequences they share. It is defined as:

$$K(x, x') = \sum \lambda^l(s, x) + l(s, x')$$

where λ is a paramter, s is a common substring, and l(s,x) is the distance between the first and the last letter of s in x (substrings need not be contiguous).

Other kernels such kernels include kernels for trees and graphs, local alignment kernels, and kernels for protein secondary structure prediction. While we chose to encode our prior knowledge into our feature sets and use the standard kernels, it would be interesting to determine the efficacy of specialized kernels trained and tested on only the sequences themselves. (9)

3 Implementation

We used the sci-kit learn python library. Sci-kit learn has an extensive SVM package and library of standard kernel functions. (10)

3.1 Feature Vectors

The following feature vectors were tested. Throughout all subsections below, we will use the following example to illustrate each idea. Sequence: TYRELDEKSNQL. The sequence length is 12, which is much smaller when compared to our data set's sequences lengths, which average around 400.

3.1.1 Feature Set 1: Amino Acid Counts (AA_counts)

Each sequence's feature vector consists of the number of occurences of each of 20 amino acids. The length of the feature vector equals 20.

Feature vector for TYRELDEKSNQL: [0,0,0,0,2,0,0,0,0,1,0,0,1,1,1,1,2,1,1,1], where there are 2 E's, 2 L's, and 1 of each other present amino acid.

3.1.2 Feature Set 2: Amino Acid Grouping Counts (AAn_counts)

Five amino acid groupings according to physiochemical properties (e.g. basic, nonbasic, polar, nonpolar) were tested . Each feacture vector consists of the number of occurences of each type of amino acid according to the current grouping scheme being used. The length equals the number of groupings for the specific grouping scheme. Note that while the order of assignment to groups has similarity implications (i.e. basic=1, nonbasic=2 would imply greater similarity than basic=1, nonbasic=4), experiments with repeated random assignment exhibited little variance.

Feature vector for TYRELDEKSNQL with n=3: [2,5,3,2]. The grouping for n=3 is as follows: AA3 = 'G':1, 'A':1, 'V':1, 'L':1, 'I':1, 'P':1, 'M':1, 'F':1, 'W':1, 'S':2, 'T':2, 'N':2, 'Q':2, 'C':2, 'Y':2, 'D':3, 'E':3, 'K':4, 'R':4, 'H':4. Amino acids that get mapped to 1 are nonpolar, those to 2 are polar, 3 are acidic (polar), and 4 are basic (polar). Our feature vector tells us that of the 12 amino acids in our sequence, 2 are nonpolar, 5 are polar, 3 are acidic, and 2 are basic.

3.1.3 Feature Set 3: Map Sequence (map_seq)

Each amino acid in the sequence is mapped to a group according to one of the grouping schemes described in feature set 2. The length equals the length of the sequence.

Feature vector for TYRELDEKSNQL with n=3: [2,2,4,3,1,3,3,4,2,2,2,1]. The same grouping is used as in the above example. Our feature vector tells us that the first amino acid in our sequence is polar, the second is polar, the third is basic, the fourth is acidic, and so on.

3.1.4 Feature Set 4: n-grams (n_grams)

Each feature vector consisted of the number of occurences of each possible n-gram. The length equals 20^n.

Feature vector for TYRELDEKSNQL with n=2: there would be a 1 corresponding to the 2-grams TY, YR, RE, EL, LD, DE, EK, SN, NQ, and QL. All other points in the length 40 vector are 0. This gives us occurences of each 2-gram through the implicit vector of all possible 2-grams [AA, AC, AD, ... , CA, CC, CD...].

3.1.5 Feature Set 5: AAn-grams (AAn_grams)

n-grams were combined with the AAn groupings in an attempt to combine sequential information with amino acid properties. Each feature vector consisted of the number of occurences of each possible AAn n-gram, where an n-gram is composed of amino acid groups. The length equals (number of groups)^n. This method also makes longer length grams computationally feasible. (8)

Feature vector for TYRELDEKSNQL with 2-grams and the 3-grouping: first, the sequence is mapped to its group representation. Just like in Feature Set 3, we get [2,2,4,3,1,3,3,4,2,2,2,1]. Next, the present 2-grams are formed. The feature vector will have a 3 for the "22" 2-gram, and 1 corresponding to the 2-grams "24", "43" "31", "13", "33", "34", "42", and "21". All other points in the length 16 vector are 0. This gives us occurences of each 2-gram through the implicit vector of all possible group-3 2-grams [11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44].

3.1.6 Feature Set 6: Amino Acid Grouping Distances(AAn_dist)

In an attempt to account for clustering of similar amino acids, this feature set counts the number of occurences of distances between successive similarly grouped amino acids. (11)

Feature vector for TYRELDEKSNQL with the 3-grouping and distance=3: starting with the basic grouping, let us rewrite the sequence as TYR**EL**D**E**KSNQL to emphasaize the basic amino acids. With this in mind, we note the first E and D are 2 apart, and the D and second E are 1 apart. Because there is 1 pair 1 apart, 1 pair 2 apart, and 0 pairs 3 apart, we get B_1=1, B_2=1, and B_3=0. This is repeated with the other groupings to get a feature vector $[P_1, P_2, P_3, NP_1, NP_2, NP_3, B_1, B_2, B_3, A_1, A_2, A_3]$.

3.1.7 Feature Set 7: Amino Acid Distances (AA_dist)

Feature set 6 can be repeated with the amino acids themselves instead of their groupings. The length equals (20) * (distances detecting).

Feature vector for TYRELDEKSNQL with distance=3: In this 60 length vector, each point will be 0 except for E_3, which equals 1.

3.1.8 Feature Set 8: Uniprot Features

This feature set uses molecular function information from the uniprot protein database, an online freely accessible database of protein sequence and function information.(13) A vector of length 54 was obtained through scraping the uniprot database for information given sequence ID's. An open source python script was adapted to glean molecular function data from the site.(4)

Feature vector for our example TYRELDEKSNQL: This is difficult to illustrate because our example is a shortened example and has no valid uniprot

id. However, let's assume that this sequence has the following functions: ATP binding, ligase activity, and phosphopantetheine binding. The feature vector would take on a 1 for the three functions and a 0 for the 51 remaining functions.

3.1.9 Feature Set 9: Alignment

In an attempt to document similarity between sequences, a global alignment suite was used to compute a pairwise alignment score of each sequence with every other sequence. If the scores were deemed higher than a certain threshold, determined by several metrics that we used, then the sequences' similarity was documented in the feature vector. The metrics used are as follows.

Let s(x_1, x_2) be the pairwise alignemnt score of sequence 1 and sequence 2. We then took the max over all comparisons as follows: similarity = $\forall_i \forall_j$ max s(x_i, x_j) - c*stdv, where c is the number of standard deviations below the max sequences are considered similar enough and stdv is the standard deviation of the all sequence comparisons. Different match, mismatch, and indel penalties were used in experimentation.

Feature vector for TYRELDEKSNQL: for the sake of this example, let's assume that there are 6 other sequences (in reality there are over 1500) in the data set. Let's say that our example is "close" given the alignment described above to the 1st, 4th, and 5th sequence. Our feature vector would be: 100110.

3.2 Kernels

Three kernels were used during testing: linear, polynomial, and radial basis function (rbf).

3.2.1 rbf

The rbf function can be defined as:

$$K(x, x') = \exp \gamma(||x - x'||_2)^2$$

where $(||x - x'||_2)^2$ is the squared Euclidean distance between the two vectors. Intuitively, gamma controls how much influence one training example extends. For instance, a smaller gamma makes the exponential curve wider, implying that its influence extends further. Note that the rbf function can be rewritten as a taylor expansion, which explains why its feature space has infinite dimension. Thus, theoretically, the rbf kernel can fully separate any dataset.

4 Results and Evaluation

4.1 Testing

We used k-fold cross validation and stratified k-fold-cross validation. Initially we used k-fold cross-validation for testing purposes because k-fold cross-validation with k=5 was used in similar research. (5) During k-fold cross-validation, the data is split evenly into 'k folds' such that one portion is used for testing purposes and the rest are used for training. Stratified k-fold cross-validation is similar to k-fold cross-validation except each fold approximates the entire dataset's distribution. In our case, this distribution refers to the substrates.

4.2 Feature Vector Results

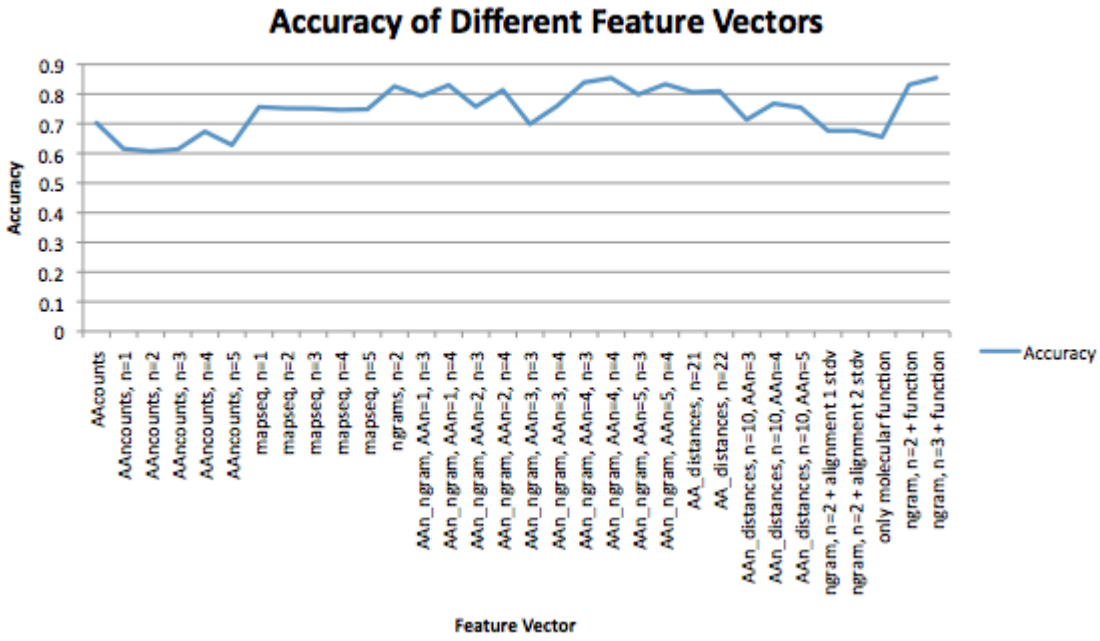


Figure 1: The following displays the accuracy of selected feature functions with a linear kernel and 5-fold-cross-validation. See appendix the for the full data.

4.3 Kernel Experimentation Results

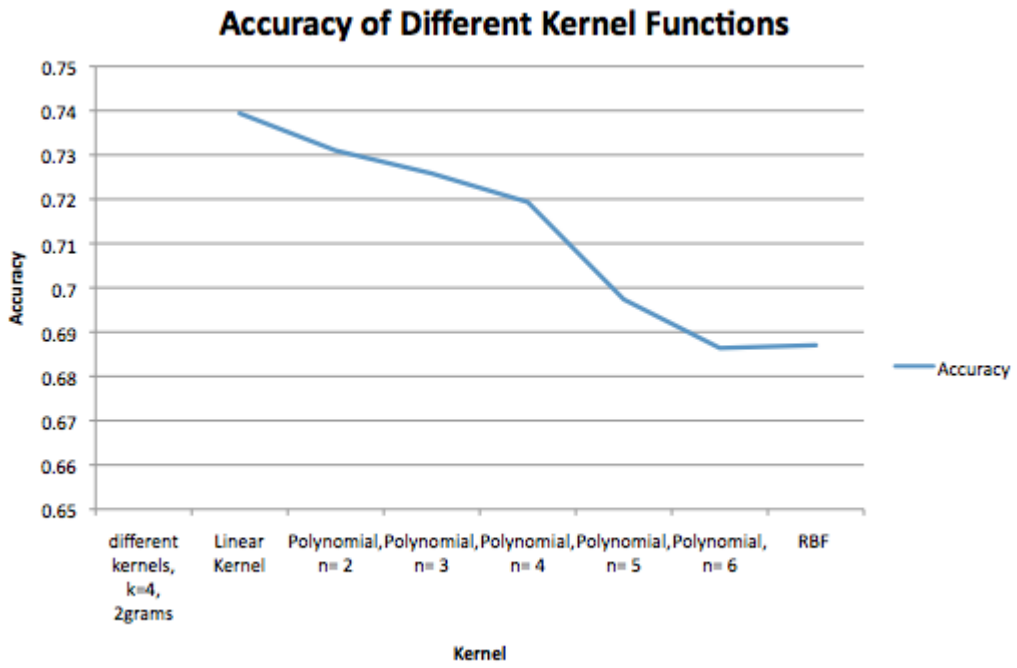
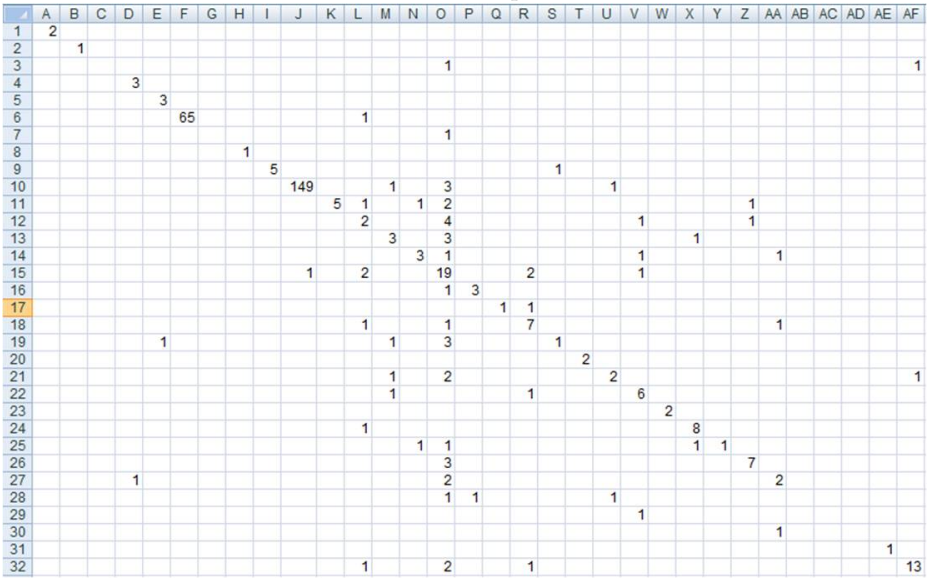


Figure 2: The following displays the accuracy of several kernels run with 2-grams and 4-fold cross-validation. See the appendix for the full data.

4.4 Confusion Matrix

Confusion matrices are used to examine if patterns in misclassifications exist.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	2																															
2		1																														
3															1																	1
4				3																												
5					3																											
6						65						1																				
7															1																	
8								1																								
9									5										1													
10										149			1		3						1											
11											5	1		1	2											1						
12												2			4							1				1						
13													3		3									1								
14														3	1							1					1					
15										1		2			19			2				1										
16															1	3																
17																	1	1														
18												1			1			7									1						
19					1								1		3				1													
20																				2												
21													1		2						2											1
22													1					1				6										
23																							2									
24												1												8								
25														1	1									1	1							
26															3											7						
27				1											2												2					
28															1	1					1											
29																						1										
30																											1					
31																															1	
32												1			2			1														13

Figure 3: The folllowing displays the confusion matrix results for testing with stratified k-fold=4, a linear kernel, and the 2-gram feature set. From the vertical line apparent in column O, we determined that many sequences were being incorrectly classified as substrate F. Sondergaard et. al faced a similar problem (5).

4.5 Unbalanced Data Set

From the confusion matrix, we were also able to determine that substrates 5 and 9 dominated the dataset. In an attempt to improve performance, we used sci-kit learn's 'class-weights' feature to manually assign penalties to misclassified sequences. This corresponds to the C misclassification error described earlier in the SVM theory section. Increasing C increases the likelihood of classifying the corresponding sequence correctly. Previously, the class-weights had been set to 'auto', which assigns weights according to the distribution of classes over the data sets. When the C values for substrates 5 and 9 were set in the range $[10^-3, 10^6]$, the accuracy varied less than 1 percent.

5 Discussion

The best performing feature vectors we observed were AAn ngrams with AAn=4 and ngram=3 concatenated with the molecular function information.

5.1 Past Work

Our results have proven more effective than the neural network approach. Sondergaard et al. received a highest accuracy of .832. Their testing was slightly different from ours because they used 5-fold cross-validation with the top 3 folds averaged toether. We used stratified 4-fold cross-validation with all folds averaged together. (5)

When HMMs were used on the same problem, however, an accuracy of .86 was obtained. We never obtained as high a mark. Perhaps if we make several of the optimizations described below and in the 'Other Topics' section, this would be possible. (5)

A discussion regarding SVM's in general is appropriate. Not unlike the neurons in neural networks, SVM's have in the past been criticized for their black box nature. This problem in SVM's arises through the kernel functions, which can often be highly unintuitive. Historically, when SVM's were first invented, many were excited by some of the technique's strong theoretical groundings. Unfortunately, the decade and a half to follow has not produced any great victories in terms of accuracy. Whether this is inherent in SVM's design is unclear, and until more experiments that illustrate which machine learning technique classifies most accurately are performed, researchers may have to continue with ad-hoc approaches.

5.2 Molecular Function

The feature vectors using molecular function in addition to n-grams proved to be the most effective. This is likely because the feature vector of only molecular function had a dimensionality of 54 whereas the other feature vectors were much larger.

In the future, molecular function could be improved as we use additional information from the uniprot database such as ligand, domain, or biological processes. Only molecular function was used in these feature vectors.

5.3 Alignment

One important note about the alignment results is that the results given are only of the first 35 sequences in the data set. This is because when ran on all sequences, the program took over 3 days, and we did not have enough computing power to finish. We ran pairwise sequence alignments comparing all sequences to all other sequences. In the future, one way to improve our results would be to run multiple sequence alignment instead of many series of pairwise alignments. Had we done this, we would likely have been able to compare all sequences and also have found more interesting similarities. Because we never ran the alignment suite on all sequences, we do not know how effective it would have been. However, the alignment suite was somewhat effective (accuracy=.7) with (match, mismatch, indel) score = (1, -2, -8) on only 35 sequences. (match, mismatch, indel) score = (1, -2, -8) proved to be more effective than (match,

mismatch, indel) score = (1, -1, -1). We could create a graph using alignment scores, with edges being present only if the similarity score was above a certain threshold, and search for cliques within this graph.

5.4 De Bruijn Graphs

Similar to alignment, De Bruijn graphs are another approach we could use to find similarities between sequences and the possible consequences on classification accuracy. Briefly, De Bruijn graphs are created by taking a number of k-mers and creating nodes and edges out of overlaps. While they are usually used in sequence assembly where the full sequence is not known, we can do the reverse and decompose our sequences into sets of k-mers.

The Hamiltonian approach sets the k-mers as nodes and creates a weighted edge between every pair of k-mers that share an overlap. The weight is the amount of overlap. Using this graph, one can attempt to find a Hamiltonian path, i.e. a path that traverses each of the nodes once. This will reconstruct the sequence. Unfortunately, finding a Hamiltonian path is NP-hard and an efficient algorithm does not exist.

Therefore, the problem is converted into finding an Eulerian path, i.e. a path that traverses each of the edges once, on a graph where the edges are the k-mers and the nodes are the overlaps. Note that there may not exist one unique Eulerian path if any substrings are repeated due to the multiplicity of loops. In our context, the ambiguity of a sequence's De Bruijn graph could serve as an extra feature. Other possibilities also exist.

6 Conclusion

Our use of Support Vector Machines in classifying A-Domain substrates has proved almost as effective as HMM's and more effective than neural networks. Perhaps if we had improved our feature vectors, SVM's would prove to be more effective than HMM's as well. While becuase of the tradeoff nature of different approaches and the differences in methodology no definitive statement can be made about SVM's versus the other machine learning approaches, our classification problem is yet another case in which SVM's have proved to be an effective means of classifying a wide variety of datasets.

7 Appendix

7.1 Table 1: Accuracy of Feature Vectors with Stratified 5-Fold Cross-Validation and a Linear Kernel.

Feature	Accuracy
AAcounts	0.702467899
AAncounts, n=1	0.615143543
AAncounts, n=2	0.606733479
AAncounts, n=3	0.613199708
AAncounts, n=4	0.673352124
AAncounts, n=5	0.628080175
mapseq, n=1	0.756137384
mapseq, n=2	0.751610815
mapseq, n=3	0.75095939
mapseq, n=4	0.746432822
mapseq, n=5	0.748376657
ngrams, n=2	0.825990187
AAn ngram, AAn=1, n=3	0.793011797
AAn ngram, AAn=1, n=4	0.829886209
AAn ngram, AAn=2, n=3	0.75743397
AAn ngram, AAn=2, n=4	0.813074434
AAn ngram, AAn=3, n=3	0.698559349
AAn ngram, AAn=3, n=4	0.759375718
AAn ngram, AAn=4, n=3	0.838928907
AAn ngram, AAn=4, n=4	0.853798935
AAn ngram, AAn=5, n=3	0.798193966
AAn ngram, AAn=5, n=4	0.83311828
AA distances, n=21	0.805950517
AA distances, n=22	0.809831924
AA distances, n=23	0.811124334
AA distances, n=24	0.809186763
AA distances, n=25	0.812427184
AA distances, n=26	0.814358493
AA distances, n=27	0.817592651
AA distances, n=28	0.82406723
AA distances, n=29	0.817592651
AA distances, n=30	0.815652991
AAn distances, n=1, AAn=1	0.631965758
AAn distances, n=1, AAn=2	0.609961374
AAn distances, n=1, AAn=3	0.608019626
AAn distances, n=1, AAn=4	0.657830671
AAn distances, n=1, AAn=5	0.634542228
AAn distances, n=2, AAn=1	0.668827644
AAn distances, n=2, AAn=2	0.646821171
AAn distances, n=2, AAn=3	0.617713749
AAn distances, n=2, AAn=4	0.699876814
AAn distances, n=2, AAn=5	0.6733563
AAn distances, n=3, AAn=1	0.698563524
AAn distances, n=3, AAn=2	0.678536382
AAn distances, n=3, AAn=3	0.650063681
AAn distances, n=3, AAn=4	0.711514772
AAn distances, n=3, AAn=5	0.703110972

7.2 Table Contd.

Feature	Accuracy
AAn distances, n=4, AAn=1	0.716026725
AAn distances, n=4, AAn=2	0.697271114
AAn distances, n=4, AAn=3	0.674636183
AAn distances, n=4, AAn=4	0.721858232
AAn distances, n=4, AAn=5	0.708934127
AAn distances, n=5, AAn=1	0.714103769
AAn distances, n=5, AAn=2	0.701165049
AAn distances, n=5, AAn=3	0.675949473
AAn distances, n=5, AAn=4	0.723156906
AAn distances, n=5, AAn=5	0.729629398
AAn distances, n=6, AAn=1	0.728336987
AAn distances, n=6, AAn=2	0.710857083
AAn distances, n=6, AAn=3	0.692761249
AAn distances, n=6, AAn=4	0.744501514
AAn distances, n=6, AAn=5	0.728977973
AAn distances, n=7, AAn=1	0.734796952
AAn distances, n=7, AAn=2	0.725090302
AAn distances, n=7, AAn=3	0.702457459
AAn distances, n=7, AAn=4	0.75033302
AAn distances, n=7, AAn=5	0.74062637
AAn distances, n=8, AAn=1	0.747092598
AAn distances, n=8, AAn=2	0.734155966
AAn distances, n=8, AAn=3	0.697924627
AAn distances, n=8, AAn=4	0.763900198
AAn distances, n=8, AAn=5	0.741269444
AAn distances, n=9, AAn=1	0.749679507
AAn distances, n=9, AAn=2	0.728975885
AAn distances, n=9, AAn=3	0.703754045
AAn distances, n=9, AAn=4	0.763902286
AAn distances, n=9, AAn=5	0.74710095
AAn distances, n=10, AAn=1	0.756790897
AAn distances, n=10, AAn=2	0.743204927
AAn distances, n=10, AAn=3	0.713448168
AAn distances, n=10, AAn=4	0.767789957
AAn distances, n=10, AAn=5	0.754203988
ngram, n=2 + alignment 1 stdv	0.67619048
ngram, n=2 + alignment 2 stdv	0.67619048
alignment, 1stdv, 1, -2, -8	0.704761905
only molecular function	0.655235411
ngram, n=2 + function	0.83116818
ngram, n=3 + function	0.854444096

7.3 Table 2: Accuracy of Various Kernel Functions with 4-Fold Cross-Validation and 2-grams

Kernel	**Accuracy**
Linear Kernel	0.739382924
Polynomial, n= 2	0.73097495
Polynomial, n= 3	0.725801971
Polynomial, n= 4	0.71934537
Polynomial, n= 5	0.697368157
Polynomial, n= 6	0.686382898
RBF	0.687027219

8 Acknowledgements

We would like to thank Professor Jotun Hein for his guidance throughout this project.

References

[1] Burges, Christopher J. C. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery* 2 (1998): 121-67.

[2] Carlos Prieto, Carlos García-Estrada, Diego Lorenzana, and Juan Francisco Martín. Nrpssp: non-ribosomal peptide synthase substrate predictor. *Bioinformatics*, 28(3):426–7, Feb 2012. doi: 10.1093/bioinformatics/btr659.

[3] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classification." National Taiwan University, 15 Apr. 2010.

[4] Dalloliogm. "Scrape Uniprot." N.p., 21 Sept. 2011. Web. 02 Dec. 2013.

[5] Dan Ariel Søndergaard, Torben Muldvang Andersen, and Henrik Schmidt-Møller. Classification of Non-Ribosomal Peptide Syntheses with Feed-Forward Neural Networks. *Challenges in Bioinformatics*, Fri. 25 Oct. 2013.

[6] Fletcher, Tristan. "SVMs Explained." University College London, n.d. Web. 02 Dec. 2013.

[7] Florian Markowetz, Lutz Edler, and Martin Vingron. "Support Vector Machines for Protein Fold Class Prediction."

[8] Hong, Huixiao, Qilong Hong, Roger Perkins, Leming Shi, Hong Fang, Zhenqiang Su, Yvonne Dragan, James C. Fuscoe, and Weida Tong. "The Accurate Prediction of Protein Family from Amino Acid Sequence by Measuring Features of Sequence Fragments." *Journal Of Computational Biology* 16.12 (2009): 1671-688.

[9] Scholkopf, Bernhard, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology*. Cambridge, MA: MIT, 2004. Print.

[10] "Scikit Learn." Machine Learning in Python. N.p., 2010. Web. 02 Dec. 2013.

[11] Setsuro Matsuda, Jean-Philippe Vert, Hiroto Saigo, Nobuhisa Ueda, Hiroyuki Toh, and Tatsuya Akutsu. "A Novel Representation of Protein Sequences for Prediction of Subcellular Location ing Support Vector Machines." *Protein Science* 14.11 (2005): 2804-813.

[12] Sewell, Martin. "VC Dimension." Support Vector Machines. University College London, Sept. 2008. Web. 2 Dec. 2013.

[13] "UniProt." Wikipedia. Wikimedia Foundation, 11 Apr. 2013. Web. 02 Dec. 2013.

[14] "UniProt." UniProt. Uniprot Consortium, 2002. Web. 01 Dec. 2013.

[15] Yoon, Byung-Jun. "Hidden Markov Models and Their Applications in Biological Sequence Analysis." Current Genomics 10.6 (2009): 402-15.