Honours Year Project Report

A Generic System for Genomic Feature Recognition

By

Koh Chuan Hock

Department of Computer Science

School of Computing

National University of Singapore

2007/2008

Abstract

Functional sites such as transcription start sites, translation initiation sites and polyadenylation sites influence virtually all aspects of the gene expression process. A general approach for computational recognition of these sites consists of feature generation, feature selection, feature integration and possibly also the construction of cascade classifiers. In this report, I have described a software tool, Sirius Prediction Systems Builder (PSB) which I have built, that supports this approach. Using PSB, I have built two prediction models, one for the recognition of Arabidopsis polyadenylation sites and another for the subcellular localization of proteins. Both systems have superseded current-state-of-art models based on evaluation of public datasets. On top of being able to produce high-quality results, the other key features of PSB lies in the fact that it is hassle-free and that the time required to build a prediction model is greatly reduced. In place of programming languages is a user-friendly graphical user interface, taking the burden of programming off users. Genetic algorithm assists in the feature generation step, and users no longer need to spend extended periods deciding on the features to generate. Prediction models built can easily be saved and reused, and can even by put online as prediction servers.

Subject Descriptors:

- I.2 Artificial Intelligence
- I.5 Pattern Recognition
- I.6 Simulation and Modeling
- J.3 Life and Medical Sciences

Keywords:

Arabidopsis, computational recognition, functional sites

Table of Contents

1. Introduction

The Human Genome Project was started formally in 1990, and one of its goals was to determine the sequences of the 3 billion chemical base pairs that make up human DNA. Due to the Human Genome Project, sequencing technologies have advanced tremendously in terms of both cost and time. From US$10 per finished base to 10 finished base per US$1 (Shendure, Mitra, Varma, and Church, 2004). From 13 years to finish the first human genome to, in 2007, sequencing James Watson's genome in just two months (Chi, 2008). Sequencing will only get cheaper and faster in the years to come.

At the moment, the full genome sequences of many different organisms are available. With these next-generation sequencing technologies, it is expected that more genome will be sequenced. Using sequencing, it is also possible to get protein sequences via mRNA. By obtaining the mRNA sequence, we can use the RNA codon table to convert it into peptide sequences. With so much data, the next challenge would be to derive meaningful knowledge from these huge amounts of data.

In particular, when given a DNA sequence, finding its functional sites such as transcription start sites, translation initiation sites, and polyadenylation sites have always been of interest to biologists as functional sites influence virtually all aspects of the gene expression process. In another example, the ability to determine the subcellular localization of the protein from a protein sequence, can give biologists clues to the functions of the protein.

Biologists would usually depend on the predictions of computer systems to determine possible locations of functional sites or possible subcellular localization of proteins. From there, they would then design experiments accordingly to validate the predictions. The computational step is important because it would greatly reduce the number of experiments that needs to be carried out subsequently. One general approach for computational recognition of functional sites has been developed to build such a prediction system. The approach consists of the following sequential steps: 1) Feature

Generation, 2) Feature Selection, 3) Feature Integration (Liu and Wong, 2003a) and 4) Cascade Classifier (Koh and Wong, 2007).

Several high quality prediction systems for functional sites have been developed using the approach (Koh et al, 2007 and Liu, Han, Li and Wong, 2005). It will also be shown in this report that this approach could also be used to build a prediction system for subcellular localization of proteins which is comparable to some well-known prediction systems for this problem.

However, each prediction system is developed by skilled programmers to solve just one particular problem. For example, Koh et al (2007) developed a system to do prediction of polyadenylation sites for Arabidopsis genomic sequences using Java programming language. Building such a system is time-consuming and requires specialized skills, and when such a system is built, it is very specific in that it carries out prediction only for a particular functional site of a particular organism. As such, the Koh et al (2007) system can neither carry out prediction for other functional sites except polyadenylation sites for Arabidopsis genomic sequences nor can it be used to do prediction for polyadenylation sites on organisms other than Arabidopsis.

Furthermore, with sequencing technologies advancing so rapidly, the rate of producing high quality prediction systems has to improve or it might become a bottleneck step. Currently, there exists no other software that supports the abovementioned approach. The closest software that lends some support to the methodology would be machine learning packages. One popular machine learning package that is often used to carry out the feature selection and feature integration step is Waikato Environment for Knowledge Analysis (WEKA). WEKA is a free machine learning software package written in Java and developed at University of Waikato (Witten and Frank, 2005).

There are two main obstacles when using WEKA to build classifiers for sequences. Firstly, as WEKA is a general machine learning package, it does not support the feature generation step for sequences and cascade classifier step, which is often used in analysing biological sequences. Therefore, many additional programs have to be

written to interface with WEKA to achieve the desired results. Secondly, WEKA does not offer straightforward ways to save and reuse trained classifiers.

In an attempt to enable prediction systems to be built, saved and used with ease and speed using the mentioned approach, I developed a software tool named Sirius Prediction System Builder (Sirius PSB). Sirius PSB exudes user-friendliness in that it is equipped with a nice Graphical User Interface that will allows anyone with just some basic knowledge in data mining to be able to build a prediction system without any programming involved.

At the moment, Sirius PSB can handle both DNA and protein sequences. It can build prediction models that make predictions for any of the following; on all position of a sequence, only when a motif is encountered, or one prediction for each sequence.

To elaborate, Sirius PSB has the ability to build prediction systems to solve several different types of sequence prediction problems. For instance, it can be used to build prediction systems for both DNA and protein sequences. For DNA sequences, it can build models that carry out predictions for functional sites with or without highly conserved motifs as anchors. In the latter, it can build models that predict the localization of the protein and it can also build models that can find the cleavage sites.

In this report, two prediction models have been built using real datasets to show the capabilities of Sirius PSB. Both models have managed to achieve results comparable to the current state of art.

2. Feature Generation, Feature Selection, Feature Integration and Cascade Classifier

In this section, we will discuss a general approach- feature generation, feature selection, feature integration and cascade classification, which have been used repeatedly to do functional sites prediction. This approach can also be easily generalized to build models to do prediction on subcellular localization of proteins.

2.1 Feature Generation

In the feature generation step, a set of candidate features are generated based on the sequences as sequences are usually not suitable to be used directly on machine learning techniques.

In this step, one common feature type that is frequently used is the k-gram feature. It is typically used because it is an easy type of feature to extract and compute. It also been shown that just by using them, high-quality classifiers can be produced (Koh et al, 2007 and Liu, Han, Li and Wong, 2003b).

A k-gram feature is simply a string of k consecutive characters and the frequency of that string in a window location. The characters are usually one of the symbols according to the IUPAC (International Union of Pure and Applied Chemistry) depending on whether the sequence of interest is a genomic or peptide sequence. Window location tells us which part of the sequence we should look at to calculate the number of occurrences of the string.

Other features that are closely related to k-gram feature are ratio of k-gram features and multiple k-gram features. Ratio of k-gram feature is where we calculate the ratio of two k-gram features in a certain window location. Multiple k-grams are two or more k-gram features that occur one after another within a specific distance range of each other in the window.

Note that in this feature generation step, it is not purely limited to k-gram feature or its closely-related counterparts. It really depends on the imagination of one to decide what type of features to extract when given a sequence.

It is not difficult to see that the feature generation step is the most critical step of all. When given a “correct” set of candidate features, one can easily build a high accuracy classifier from them. However, the task of finding the “correct” set of candidate features is similar to the motif finding problem given a set of sequences, which is known to be NP-hard.

2.2 Feature Selection

Usually, in the feature generation step, a large set of candidate features will be generated. This will pose two immediate problems. Firstly, with too many candidate features comes the curse of dimensionality. Secondly, many of these features could possibly be noise or irrelevant features. Having such features for training of most machine learning techniques would often lead to over fitting.

Therefore, it is with hope that by employing this feature selection step, noise and irrelevant features will be severely reduced. Various techniques may be used to carry out the differentiation between meaningful and useless features. Techniques like signal-to-noise measure, statistical measure, entropy measure, information gain measure, correlation-based measure and so forth can all be used for this cause.

2.3 Feature Integration

After feature selection, those features that have selected will be used to train a machine learning method. This trained feature integration classifier is then ready to be used.

For sequence prediction, there are generally three different categories. One is

where prediction will only be made once for each sequence (e.g. subcellular localization of proteins). Another would be where there exists known anchor motif and prediction would only be made when the anchor motif is encountered. The last type is where there does not exist any anchor motif and every position of the sequence is a candidate site and prediction has to be made on every position of the sequence.

For the first two types of problems, simply using the first 3 step of the approach is sufficient to build decent prediction models. For the last category, an additional step is employed, that is, cascade classifier.

2.4 Cascade Classifier

The reason for an additional step is because previously, problems that have been tackled using the feature generation, feature selection and feature integration always have anchor points to base the predictions upon. For example, Liu et al (2003b) does prediction of polyadenylation sites of humans. The authors would first scan for a AATAAA motif in the sequence and then predict if that is a true polyadenylation site. However, using such a scanning method has a disadvantage, that is, although 58.2% of human polyadenylation site contains an AATAAA upstream, there are motifs other than AATAAA can also be a true polyadenylation site (Beaudoing, Freier, Wyatt, Claverie and Gautheret, 2000). Therefore, the prediction model will surely fail to recognize those sites.

Also, not all functional sites have an anchor motif. For example, the most frequently occurring motif (AATAAA) for Arabidopsis polyadenylation sites is found only in about 10% of Arabidopsis genes (Loke, Stahlberg, Strenski, Haas, Wood and Li, 2005).

For sequences without an anchor motif, feature integration will first be used to create an output with a prediction score for each position on the sequence. Thereafter, these prediction scores will be used as a feature to train a cascade classifier using machine learning. Any machine learning technique that can handle numerical features can be used here.

3. Sirius Prediction System Builder

This software tool aims to assist anyone with some basic knowledge in data mining to be able to build, save and use a classifier in a hassle-free and fast way using the feature generation, feature selection, feature integration and possibly cascade classification approach. Sirius, which is also commonly known as the "Dog Star", is based on the fact that I am born also in the year of Dog according to lunar calendar, which is why I have decided to name the software tool Sirius Prediction System Builder.

3.1 Overview

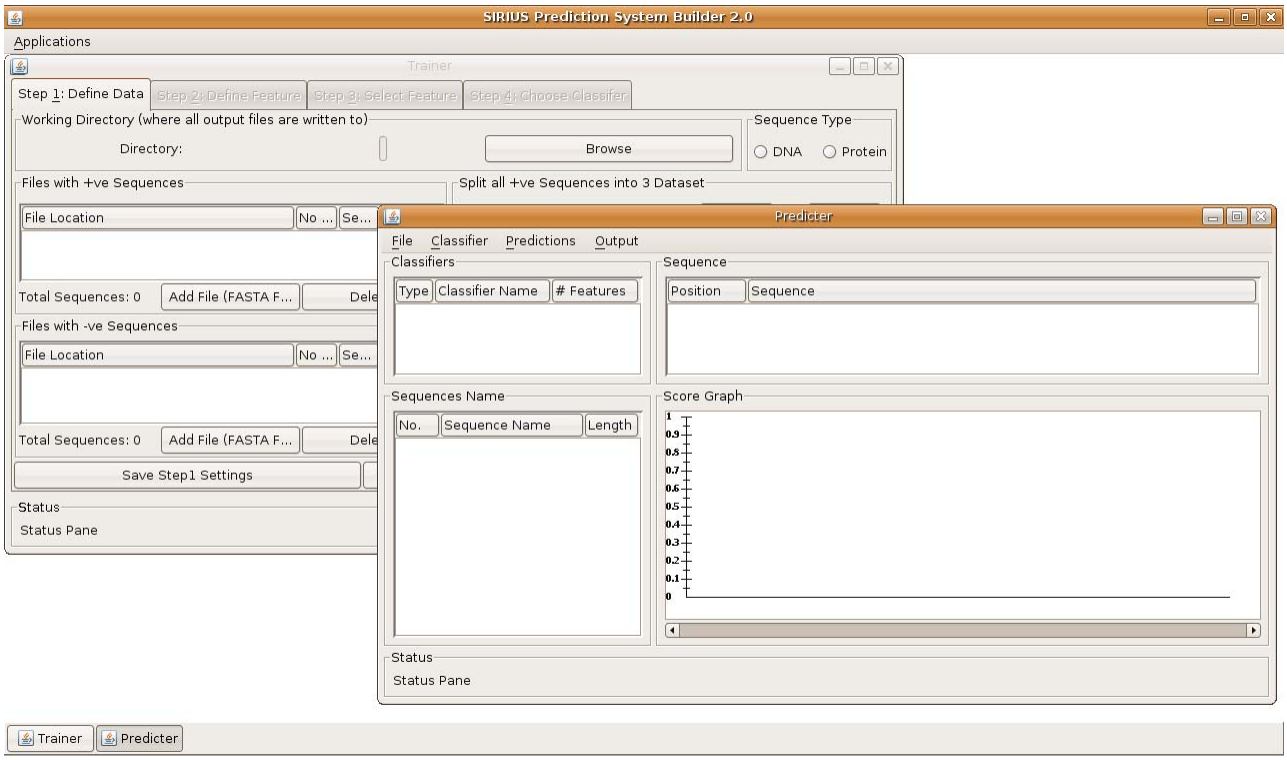

Figure 1. Overview of Sirius PSB

Sirius PSB consists of two applications namely Trainer and Predictor. Trainer serves to build a classifier from scratch and all the user has to do is to prepare sequences which typically contain a set of “positive” sequences and a set of “negative” sequences. This can often be done using publicly available database servers, or obtain datasets that are already prepared by others. “Positive” sequences here refer to a set of known related sequences.

For example, if the user is interested in subcellular localization of proteins, a set of protein sequences that are known to be found in mitochondria would be the “positive” sequences whereas the “negative” would be any protein sequences that are known *not* to be found in mitochondria.

Predictor allows users to use classifiers that they have saved previously using Trainer. After loading the classifier, users can load in new sequences and do prediction on them.

The following sections serve to describe the basic features and some capabilities of Sirius PSB rather than a step-by-step user manual guide. A step-by-step user manual guide is currently unavailable. However, users who are familiar with WEKA should not encounter much difficulty while using Sirius PSB.

3.2 Trainer

The usage of Trainer is straightforward as it will be guided using 4 different tabs. Each tab consists of something that the user needs to do before proceeding on to the next tab.

Although WEKA has its limitations in supporting the approach mentioned, but it can support part of the approach - feature selection and feature integration. Furthermore, it is a popular and powerful general machine learning tool which is familiar to many. Hence, incorporating it would make learning and using Sirius PSB much easier. In Trainer, some of the capabilities of WEKA are incorporated into it.

3.2.1 Step 1: Define Data

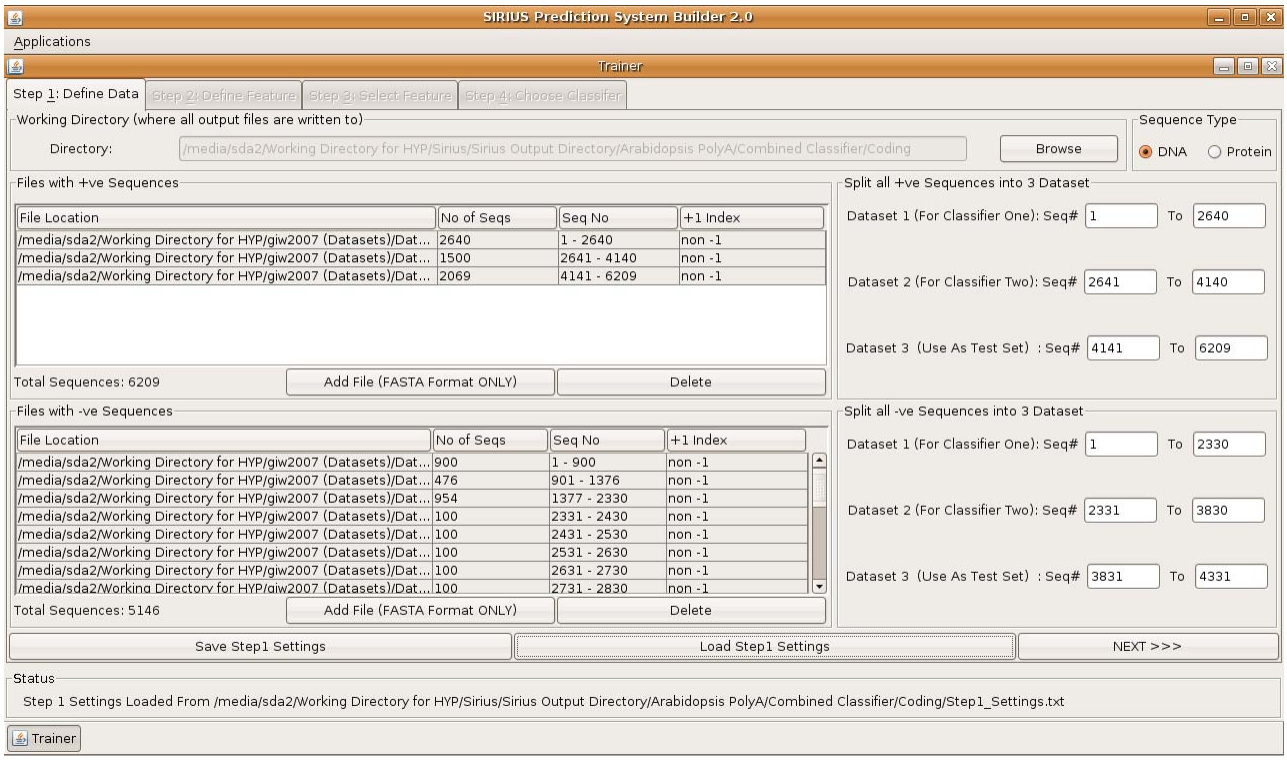

Figure 2. 'Step1: Define Data' tab

In this tab, the user should first decide on a working directory where all the output files will be saved. The user will then have to direct Sirius PSB as to where the "positive" and "negative" sequences can be obtained. All sequences are to be in FASTA format. Sirius PSB needs to know if the sequences are that of DNA or proteins by having the user select the corresponding radio button. Finally, the user has to tell Sirius PSB how the sequences are to be used. Dataset 1 would be used for training of Feature Integration classifier. Dataset 2 would be used for training of Cascade Classifier. Dataset 3 would be used as blind test set.

As there is quite a bit of information to feed to Sirius PSB, the user also has the option to save the settings so that he can skip this step the next time. The user can then click on the "Next" button to proceed to next step.

3.2.2 Step 2: Define Feature

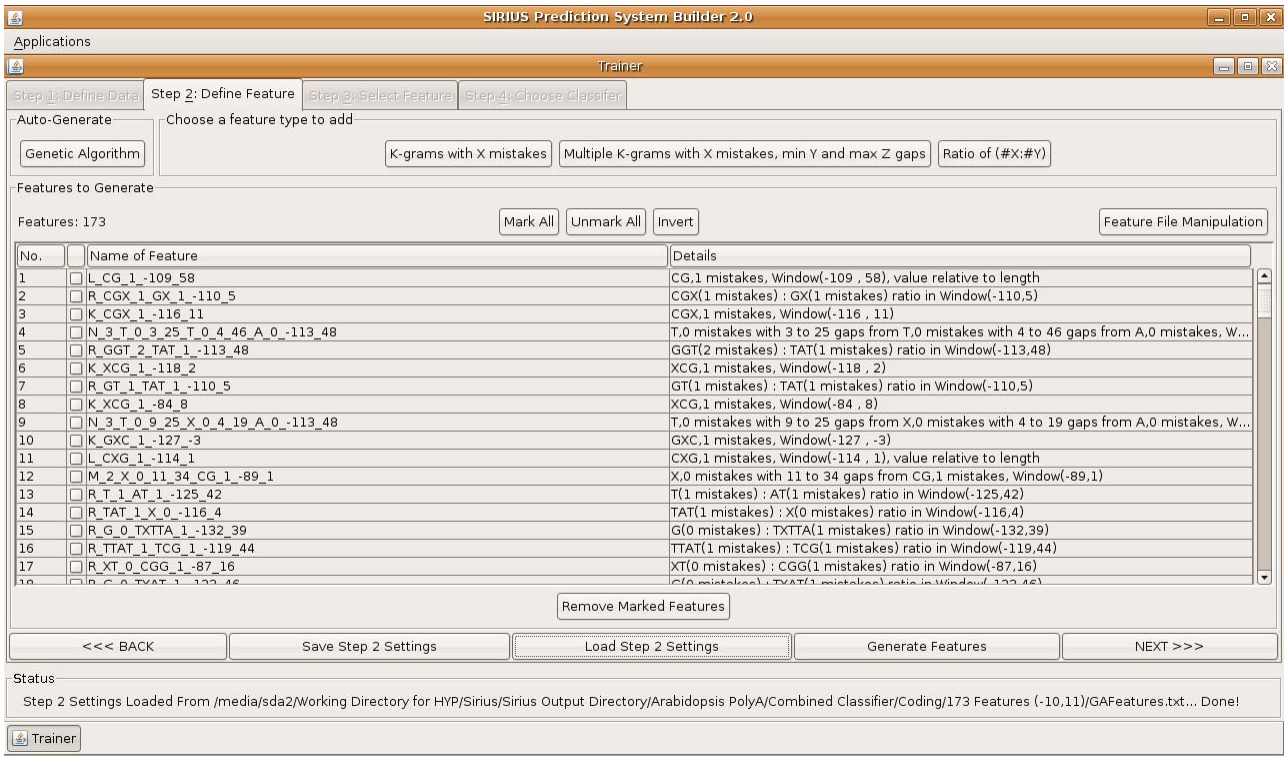

Figure 3. 'Step 2: Define Feature' tab

This tab corresponds to the Feature Generation step of the approach. Therefore, in this tab, the user basically has to decide upon the candidate features to generate. The 3 different types of features that are currently supported by Sirius PSB are k-grams feature, multiple k-gram feature and ratio of k-gram feature.

For each of them, the user can choose to generate all permutations given a fixed window or just one particular instance. For example, figure X depicts how the user can generate all the possible permutations of 3-gram for DNA sequence with the A, C, T, G characters with window location being (-30, 30). Once the user click the "OK" button, 4^3 = 64 different k-gram features will be generated.

The user can then choose to remove some features using the "Remove Marked Features" button if there are any features the user decides should not be included in the

set of candidate features.

As discussed in section 2.1, users may not always know what features to generate, though granted, users can look up literature for the more prominent motifs. However, there may be times where literature on the particular topic may be limited. Therefore, I have implemented a genetic algorithm in attempt to mine interesting features from the training sequences provided by user. (More details of the implemented Genetic Algorithm can be found in Appendix A)

Likewise, in this step, the user can also choose to save the settings, which is the set of candidate features that the user has generated for future reuse.

After finalising the set of candidate features that the user is interested in, the user should click on “Generate Feature” button to do computation of the features for Dataset 1. Once the computation is done, the screen will be switched to the next step.

3.2.3 Step 3: Select Feature

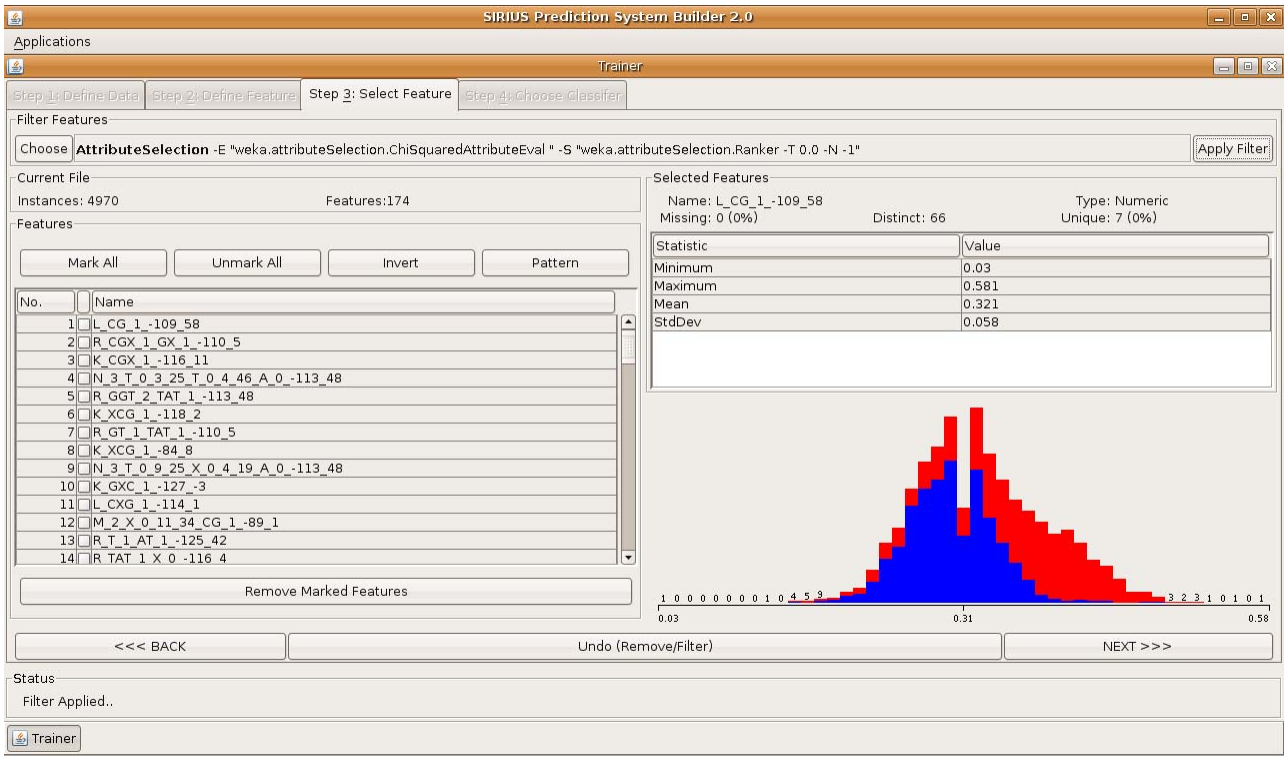

Figure 4. 'Step 3: Select Feature' tab

This tab is where Sirius PSB carries out feature selection. As mentioned earlier, this step uses some APIs provided by WEKA. Therefore, Sirius PSB is not limited to only a few feature selection methods as it contains all the feature selection methods available in WEKA. Furthermore, it will be automatically upgraded with each newer version of WEKA as Sirius PSB calls for WEKA APIs instead of embedding them. Sirius PSB also displays the class distribution graph and some statistics of each feature which are also extracted from WEKA.

Note that the interface of Sirius PSB in this step is a deliberate attempt in mimicking that of WEKA so that users who are familiar with WEKA will feel more at ease using Sirius PSB.

After the user is done with feature selection on the set of candidate features, the

user can click on the “Next” button to move on to the final tab of Trainer.

3.2.4 Step 4: Choose Classifier

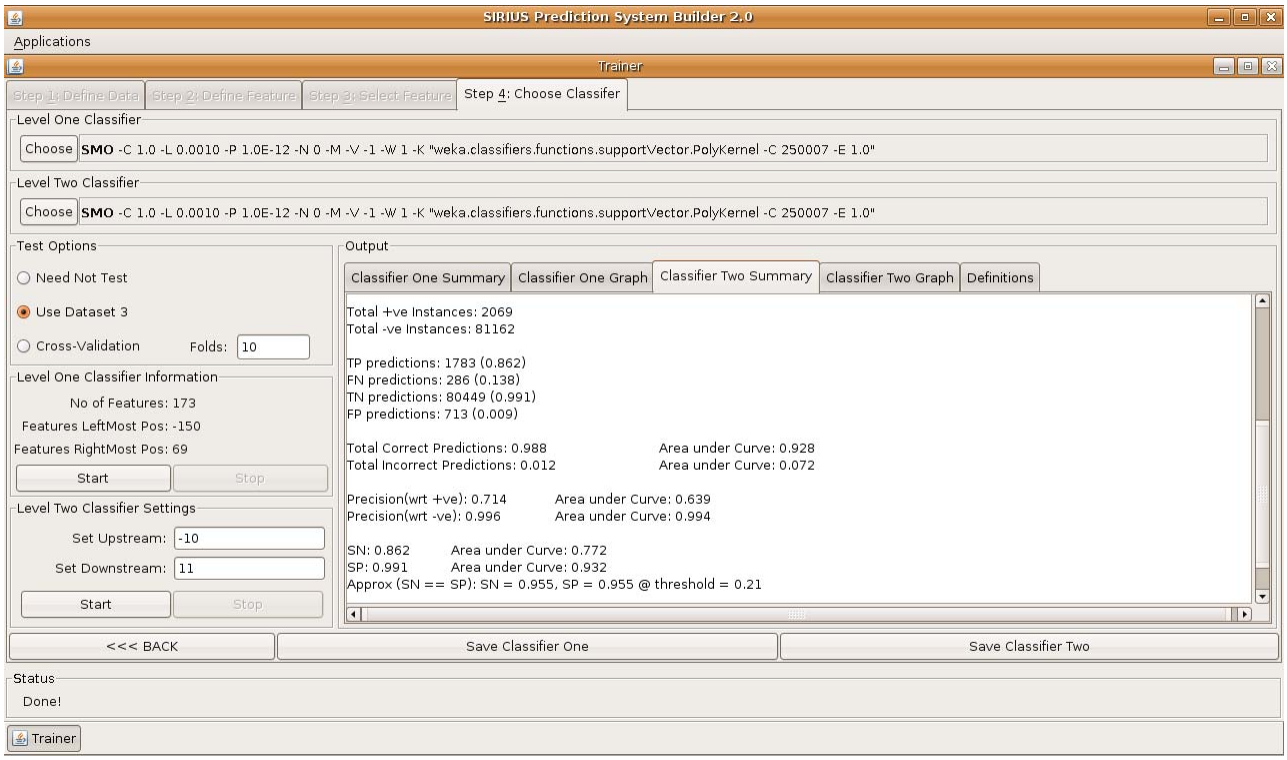

Figure 5. ‘Step 4: Choose Classifier’ tab – Performance Measures Summary

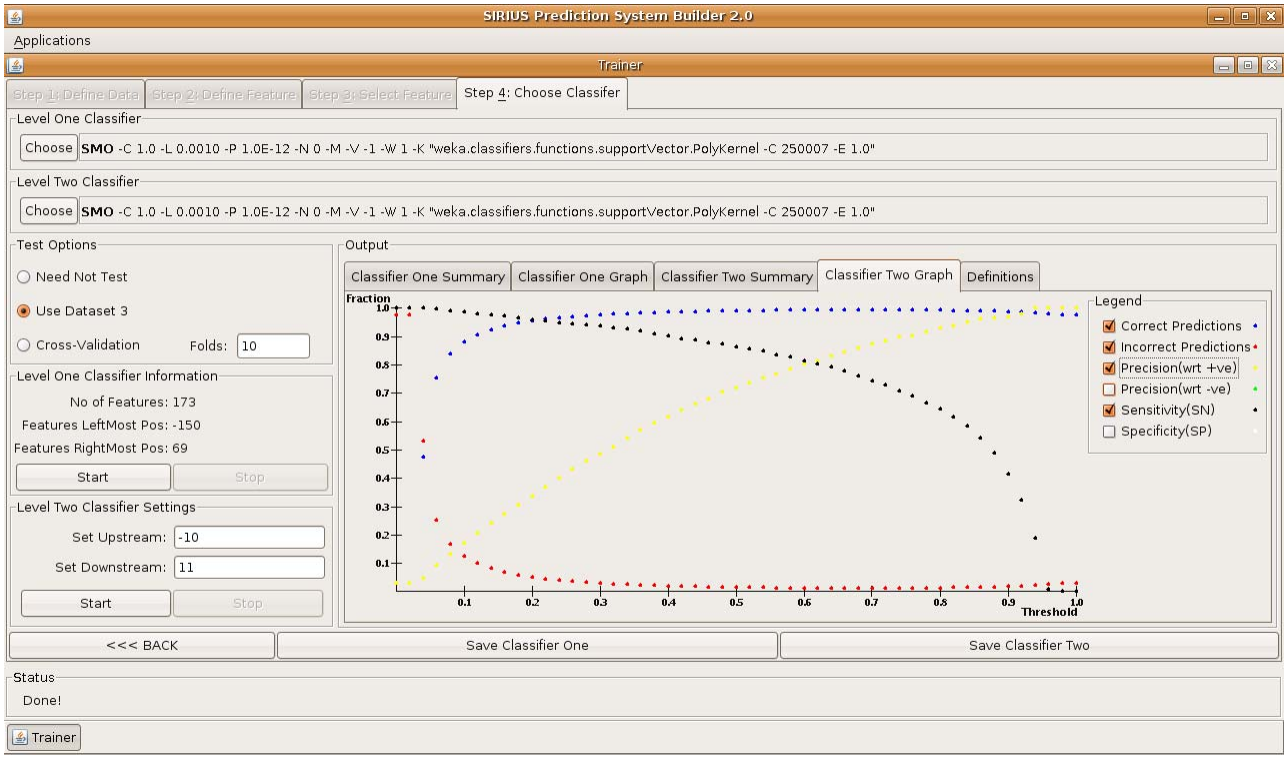

Figure 6. ‘Step 4: Choose Classifier’ tab – Graph

What occurs in this tab is the encompassing of the feature integration and cascade classification step of the approach. Here, level one classifier refers to the feature integration classifier while level two classifier refers to the cascade classifier. Again, as Sirius PSB uses APIs directly from WEKA, it offers all the machine learning techniques that WEKA offers. As before, it will be also automatically upgraded together with newer releases of WEKA as Sirius PSB calls for WEKA APIs instead of embedding them.

In this tab, there are options to either use cross-validation or blind test set (Dataset 3) to test for the accuracy of the trained classifiers. The user also has the option of forgoing the testing. If users choose to test the trained classifiers, either by using cross-validation or blind test set, a comprehensive summary of the classifier performance will be displayed (refer to Figure 5). Furthermore, there is also a graph showing the fraction vs. threshold of various performance measurement (refer to Figure 6).

After training the classifiers, the user can then choose to save them. The saved classifiers can later be used in Predictor or set up as a prediction server.

3.3 Predictor

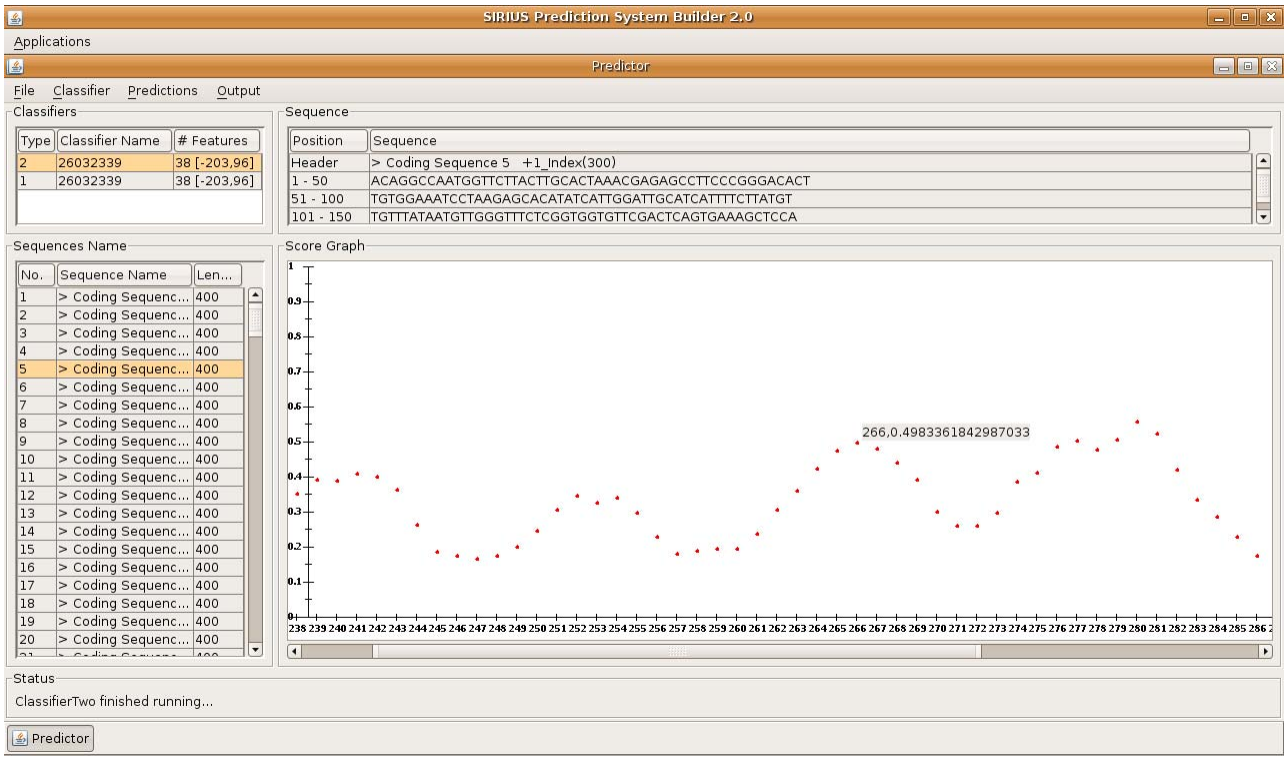

Figure 7. Predictor

Predictor provides the capabilities to load a classifier and run it on sequences to generate prediction scores. After loading a classifier, the user can load in a file of sequences stored in FASTA format. Once a classifier had been run on a set of sequences, Predictor will graphically display the prediction scores of the classifier on the sequences.

The user also has the option to decide on whether to make prediction on all positions of the sequence or only on user-specified motifs. The all positions mode would be useful when each position of the sequence is a possible candidate site. The user-specified motifs mode would come in handy when the user knows that only certain sequences can be the 'real' site.

4. Prediction Systems built using Sirius PSB

After creating Sirius PSB, the next obvious step would be to demonstrate that Sirius PSB is indeed capable of producing decent prediction systems for real-life applications. Therefore, I have built two prediction models using Sirius PSB. One is for prediction of subcellular localization of proteins and the other, for recognition of Arabidopsis polyadenylation site. It took me approximately one week to build them.

4.1. Subcellular Localization of Proteins

4.1.1 Introduction

Given a protein sequence, it would be interesting to know the subcellular localization of the protein as it can allow us to better understand its function. Many prediction models have been constructed previously to predict a protein subcellular localization based on its sequence. In particular, TargetP (Emanuelsson, Nielsen, Brunak and Heijne, 2000) is one such model. It has achieved a high sensitivity (>85%) and is still often used by biologists till today. Hence, I will compare my protein localization model against TargetP.

4.1.2 Datasets

The dataset used here is downloaded from the TargetP website. All sequences were extracted from SWISS-PROT and redundancy reduced. Please refer to Emanuelsson et al (2000) for more details on the preparation of the dataset.

The dataset has two version, plant and non-plant. For the plant version, it contains 141 cTP, 368 mTP, 269 SP and 162 “other” sequences. For the non-plant version, it contains 371 mTP, 715 SP and 1652 “other” sequences.

The abbreviations used subsequently are as follows: cTP stands for chloroplast transit peptides, mTP stands for mitochondrial targeting peptides, SP stands for signal peptides and “other” stands for peptides in other localizations.

4.1.3 Methods

TargetP is built using neural networks and consists of two layers. The first layer is a dedicated network for each presequence (cTP, mTP, SP), and the second layer is an integrating network that outputs the actual prediction. A non-plant version of TargetP that distinguishes only between mTP, SP and “other” has also been constructed.

For my protein localization model (PL model), I employed the first 3 step of the approach- Feature generation, feature selection and feature integration.

For the feature generation step, I used straightforward features of 1, 2 and 3-gram with window (0,100). Note that there are 20 different amino acids. This means I have $20 + 20^2 + 20^3 = 8420$ features. I will then calculate the occurrence of the 8420 features for the first 101 characters of each sequence. For the feature selection step, I filtered away those with chi-square value <= 0. As for the feature integration step, I used Support Vector Machine (Cortes and Vapnik, 1995) with polynomial kernel of degree two. Like TargetP, I also have seven different SVM classifiers for each type of presequence.

4.1.4 Results

Table 1. Prediction performance based on 5-fold cross-validation of TargetP, PL model and Upgraded PL model.

			TargetP			PL model			Upgraded PL model		
Set	Category	Size	TP	FN	SN	TP	FN	SN	TP	FN	SN
Plant	cTP	141	120	21	0.851	127	14	0.901			
	mTP	368	300	68	0.815	297	71	0.807			
	SP	269	245	15	0.911	253	16	0.941			
	other	162	137	25	0.846	142	20	0.877			
Plant Sensitivity			0.856			**0.882**					

Non-plant	mTP	371	330	41	0.889	344	27	0.927			
	SP	715	683	32	0.955	204	511	0.285	466	79	0.855
	other	1652	1451	201	0.878	1552	100	0.939			
Non-plant Sensitivity			**0.907**			0.717			**0.907**		
Overall Sensitivity			0.878			0.811			**0.892**		

The results for TargetP are extracted from Emanuelsson et al (2000). Since the authors used 5-fold cross-validation, in order to compare with TargetP, I also ran 5-fold cross-validation on PL model and Upgraded PL model.

As the initial results of the SVM classifier for non-plant SP sequence were not satisfactory, I re-trained it using new features generated from genetic algorithm. The only difference between Upgraded PL model and PL model are the features used to train non-plant SP SVM classifier.

170 sequences from non-plant SP were randomly chosen to run genetic algorithm. Therefore, the Upgraded PL model only used 545 (715 – 170) sequences for the cross-validation to prevent obtaining over-optimistic results.

4.1.5 Discussion

From the results, it is clear that using the approach (feature generation, feature selection and feature integration) is able to obtain comparable if not better performance for the prediction of subcellular localization of proteins.

It is not surprising to see why PL model is able to achieve good results simply based on the features that only consider the first 101 positions of each sequences, as biological facts tell us that the first part of protein sequences contains its “address”.

One highly probable reason as to why PL model does not perform up to par on non-plant SP sequences could be that those sequences do not have any distinguishing

characteristics in the first 101 positions to allow the classifier to differentiate non-plant SP sequences from the rest.

After running genetic algorithm, I simply selected the top 120 features from the set of features generated after some filtering. With only these 120 features, Upgraded PL model was able to produce a much improved classifier. This shows that the implemented genetic algorithm indeed is able to mine “interesting” features.

Finally, all this was done in simply one week using Sirius PSB. During the one week, I was also concurrently working on building a prediction model for polyadenylation sites of Arabidopsis, which I will discuss in the next section.

4.2 Recognition of Polyadenylation Sites from Arabidopsis Genomic Sequences

4.2.1 Introduction

Polyadenylation is a post-transcriptional process. The process basically cleaves and adds approximately 200-300 adenosine residues to the pre-mRNA 3' end. This process has been shown to be an essential processing event and an integral part of gene expression (Loke et al, 2005). Having the ability to accurately predict them allows us to define gene boundaries, predict the number of genes as well as better understand the process.

Currently, the best prediction model for recognition of polyadenylation site for Arabidopsis sequences is designed by Koh et al (2007). Therefore, I will compare my Arabidopsis polyadenylation site model (APS model) against Koh et al (2007) model.

4.2.2 Datasets

The dataset used here are provided by Qingshun Quinn Li (Ji et al, 2007). For any two sequences with more than 70% similarity using pair-wise global alignment, one is removed. After redundancy is reduced, the dataset contains 6209 sequences with EST-supported polyadenylation sites, 1501 coding region sequences, 864 5'UTR region sequences and 1581 intronic region sequences. Each sequence is of length 400 and for the EST-supported sequences; the polyadenylation site is at position 301.The dataset are split and used in the following ways:

Dataset A (Used for training Feature Integration Classifier)

- 2640 (+ve) sequences with EST-supported polyadenylation sites
- 900 (-ve) coding region sequences
- 476 (-ve) 5'UTR region sequences
- 954 (-ve) intronic region sequences

Dataset B (Used for training Cascade Classifier)

- 1500 (+ve) sequences with EST-supported polyadenylation sites
- 100 (-ve) coding region sequences
- 100 (-ve) 5'UTR region sequences

100 (-ve) intronic region sequences

Dataset C (Used for testing)

2069 (+ve) sequences with EST-supported polyadenylation sites

501 (-ve) coding region sequences

288 (-ve) 5'UTR region sequences

527 (-ve) intronic region sequences

Both models use the same number of sequences in the same way. This is possible because both Koh et al (2007) model and APS model follow the same general approach.

4.2.3 Methods

Feature generation, feature selection, feature integration and cascade classification is the methodology used by both Koh et al (2007) model and APS model. The settings for feature selection (chi-square with threshold 0) and feature integration (support vector machine) for both models are the same. The difference between the two models is in the feature generation and cascade classification steps.

Koh et al (2007) model generates 261 candidate features based on biological knowledge from literature and uses 81 scores output by feature integration classifier (-40,41) relative to a candidate site for cascade classification. APS model generates 173 candidate features based on running genetic algorithm on Dataset A and uses (-10, 11).

4.2.4 Results

Table 2. Equal-error-rate of Koh et al (2007) model and APS model.

Control Sequences		Koh et al (2007) model Sensitivity & Specificity	APS model Sensitivity & Specificity
Coding	SN_0	0.943	**0.955**
	SN_10	0.965	**0.971**
	SN_30	0.975	**0.978**
5'UTR	SN_0	0.849	**0.854**
	SN_10	0.892	0.891

	SN_30	0.915	0.912
Intronic	SN_0	0.711	**0.724**
	SN_10	0.788	**0.791**
	SN_30	0.830	**0.833**

The performance measure used is equal-error-rate value (i.e. the points where sensitivity = specificity).

$$\text{Sensitivity (SN)} = TP/ (TP + FN)$$

$$\text{Specificity (SP)} = TN / (TN + FP)$$

where TP (True Positive) is the total number of EST-supported polyadenylation sites that are correctly predicted. FN (False Negative) is the total number of EST-supported polyadenylation sites that are not identified. TN (True Negative) is the total number of sites with prediction score <= threshold in the (-ve) sequences. FP (False Positive) is the total number of sites with score > threshold in the (-ve) sequences.

SN_0 means that the predicted polyadenylation site is exactly the same as the EST-supported polyadenylation site. SN_10 means the EST-supported polyadenylation site is within 10 nucleotides of the predicted polyadenylation site. SN_30 means the EST-supported polyadenylation site is within 30 nucleotides of the predicted polyadenylation site.

4.2.5 Discussion

From the results, APS model has shown slightly better performance over Koh et al (2007) model. What I would like to stress here is that even though both methods followed the same approach, but Koh et al (2007) model was designed by writing many programs and having to change the codes in the programs whenever the authors wanted to try different settings or generate different features. In contrast, APS model was produced by using Sirius PSB and settings can be changed simply by a few mouse clicks using the Graphical User Interface of Sirius PSB.

Another important difference is that the 261 candidate features generated by Koh

et al (2007) are decided upon after the authors spent a lot of time and effort searching and reading literature about Arabidopsis polyadenylation process. Compare this with APS model that generated the 173 features simply by running the genetic algorithm (provided by Sirius PSB) using Dataset A. This shows that with Sirius PSB's genetic algorithm, prior knowledge about the dataset is not required.

Due to these differences, Koh et al (2007) model took the authors about nine months to complete whereas APS model took me less than one week.

5. Conclusion

In this report, I have described a software tool named Sirius Prediction System Builder. Sirius PSB aims to allow users to be able to build high-quality prediction models using the feature generation, feature selection, feature integration and cascade classification approach in a manner that is hassle-free yet rapid.

Having the easy-to-use graphical user interface, even people without prior programming knowledge would be able to build a prediction model. As demonstrated using the two prediction models I have built using Sirius PSB, not only can those prediction models match current-state-of-art models in terms of accuracy, but the time required to build them is also significantly reduced.

Furthermore, with the genetic algorithm provided in Sirius PSB, users need not even worry about what features to generate. As I have shown that the genetic algorithm is able to generate “useful” features where excellent prediction models can be subsequently built from them.

With Sirius PSB, I am confident that more high-quality prediction models will be produced using the feature generation, feature selection, feature integration and cascade classification methodology. These prediction models built can easily be saved and reused, and can even by put online as prediction servers.

References

Beaudoing, E., Freier, S., Wyatt, J.R., Claverie, D.G. and Gautheret, D. (2000). Patterns of Variant Polyadenylation Signal Usage in Human Genes. Genome Research, Vol.10, No.7, July 2000, pp. 1001-1010.

Chi, K.R. (2008). The year of sequencing. Nature Methods, Vol.5, No.1, January 2008, pp. 11-14.

Cortes C. and Vapnik, V. (1995). Support-Vector Networks. Machine Learning, Vol.20, No.3, March 1995, pp.273-297.

Emanuelsson, O., Nielsen, Henrik., Brunak S. and Heijne, G.V. (2000). Predicting Subcellular Localization of Proteins Based on Their N-Terminal Amino Acid Sequence. Journal of Molecular Biology, Vol.300, May 2000, pp. 1005-1016.

Ji, G., Zheng, J., Shen, Y., Wu, X., Jiang, R., Lin, Y., Loke, J.C., Davis, K.M., Reese, G.J. and Li, Q.Q. (2007). Predictive Modeling of Plant Messenger RNA Polyadenylation Sites. BMC Bioinformatics, Vol.8, No.43, February 2007.

Koh, C.H. and Wong, L. (2007). Recognition of Polyadenylation Sites from Arabidopsis Genomic Sequences. In Proceedings of 18th International Conference on Genome Informatics, (Singapore, December 3 – 5, 2007.), pp. 73-82

Liu, H. and Wong, L. (2003a). Data Mining Tools for Biological Sequences. Journal of Bioinformatics and Computational Biology, Vol.1, No.1, April 2003, pp. 139-167.

Liu, H., Han, H., Li, J. and Wong, L. (2003b). An In-Silico Method for Prediction of Polyadenylation Signals in Human Sequences. In Proceedings of 14th International Conference on Genome Informatics, (Yokohama, December 2003), pp. 84-93.

Liu, H., Han, H., Li, J. and Wong L. (2004). Using Amino Acid Patterns to Accurately Predict Translation Initiation Sites. In silico Biology, Vol.4, No.3, March 2004, pp. 255-269.

Liu, H., Han, H., Li, J. and Wong, L. (2005). DNAFSMiner: A Web-Based Software Toolbox to Recognize Two Types of Functional Sites in DNA Sequences. Bioinformatics, Vol.21, pp. 671-673.

Loke, C.J., Stahlberg, E.A., Strenski, D.G. Haas, B.J. Wood, P.C. and Li, Q.Q. (2005). Compilation of mRNA Polyadenylation Signals in Arabidopsis Revealed a New Signal Element and Potential Secondary Structures. Plant Physiology, Vol.138, July 2005, pp. 1457-1468.

Prlic, A., Domingues, F.S. and Sippl M.J. (2000). Structure-Derived Substitution Matrices

for Alignment of Distantly Related Sequences. Protein Engineering, Vol.13, No.8, June 2000, pp. 545-550.

Shendure, J., Mitra, R.D., Varma, C. and Church G.M. (2004). Advanced Sequencing Technologies: Methods and Goals. Nature Reviews Genetics, Vol.5, May 2004, pp. 335-344.

Witten, I.H. and Frank, E. (2005). Data Mining: Practical Machine Learning Tools and Techniques, Second Edition, Morgan Kaufmann, San Francisco, 2005.

Appendix A - Genetic Algorithm

I have decided to use genetic algorithm to search the enormous possible feature space mainly due to two reasons. Firstly, genetic algorithm has been proven in many other applications to have produced excellent results (Bartels et al, 2000, Preble, Lipson and Lipson, 2005, Sam and Mani, 1996). Secondly, it is a highly flexible search algorithm. At any stage of the algorithm, you can stop it, evaluate the current features and then continue running the algorithm later with new features or adjusted settings.

1. Initialize Population
    - Each individual is a feature (K-gram, Ratio of K-gram or Multiple K-gram)
    - Unless the user provides the initial population, it will be randomly generated
    - K-gram features consists of two parts, (k-gram and x mistake allowed) and (window location)
    - Ratio of K-gram features basically consists of three parts, (k-gram and x mistakes allowed), (l-gram and y mistakes allowed) and (window location)
    - Multiple K-gram features consists of one (window location) and at least two or more (k_i-gram and x_i mistakes allowed) and (min y_i-gap and max z_i-gap)
2. Evaluation
    - The fitness score used is the chi-square value of each feature calculated based on Dataset 1 sequences
3. Selection
    - Select (selection percent (user-specified) / 100 * population size (user-specified)) into the next generation directly
    - Features are selected with a probability directly proportionate to its chi-square value
4. Crossover
    - Do (100 - selection percent) * (population size / 2) number of crossover

- Select two different features as parents
- Again, the feature with the higher score has a higher probability to be selected as a parent
- Generate two offspring (by taking parts from the parents) which will be added to the next generation

  Example:

  One k-gram feature (AACTGGA with 2 mistakes allowed) and (Window Location: -110 to 5) and one multiple of k-gram feature (ACTG with 0 mistakes allowed), (Gap: 10 to 30), (AXTTTT with 0 mistakes allowed) and (Window Location: 30 to 90) are selected as parents.

  Two possible offspring that could be generated are ratio k-gram features (AACTGGA with 2 mistakes allowed), (ACTG with 0 mistakes allowed) and (Window Location: 30 to 90) and k-gram features (ACTG with 0 mistakes allowed) and (Window Location: -110 to 5).
- Offspring generate here is ensured to be different from the parent so as to allow for diversity in the population

5. Mutation
   - Select ((mutation percent (user-specified) / 100) * population size) to do point mutation randomly
   - Selection of features are done randomly
   - Then randomly choose a part of the feature and mutate (change) one of the values

     Example:

     K-gram feature (AACTGGA with 2 mistakes allowed) and (Window Location: -110 to 5) will possibly be mutate into (AACTGG with 2 mistakes allowed) and (Window Location: -110 to 5) or (AACTGGA with 0 mistakes allowed) and (Window Location: -110 to 5) or (AACTGGA with 2 mistakes allowed) and (Window Location: -110 to -20) or (AACTGGA with 2 mistakes allowed) and (Window Location: -70 to 5)

6. Eliminate Similar Features
    - Eliminate features that are highly similar (>80%) within the population. This is to prevent population being dominated by similar features
    - Features are (>80%) similar if their window location overlaps by more than 80% and when the rest of the parts are almost identical
7. Replenish
    - Generate new features randomly to replenish the population to the desired size
8. Return to step 2 unless termination generation (user-specified) is reached