Journal of Bioinformatics and Computational Biology
Vol. 9, No. 3 (2011) 399–413
© Imperial College Press
DOI: 10.1142/S0219720011005586

Imperial College Press
www.icpress.co.uk

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

# A TWO-STAGE EVOLUTIONARY APPROACH FOR EFFECTIVE CLASSIFICATION OF HYPERSENSITIVE DNA SEQUENCES

UDAY KAMATH*, AMARDA SHEHU† and KENNETH A. DE JONG‡

*Department of Computer Science, George Mason University*
*Fairfax, Virginia 20123, USA*
**ukamath@gmu.edu*
†*ashehu@gmu.edu*
‡*kdejong@gmu.edu*

Received 31 January 2011
Revised 6 April 2011
Accepted 14 April 2011

Hypersensitive (HS) sites in genomic sequences are reliable markers of DNA regulatory regions that control gene expression. Annotation of regulatory regions is important in understanding phenotypical differences among cells and diseases linked to pathologies in protein expression. Several computational techniques are devoted to mapping out regulatory regions in DNA by initially identifying HS sequences. Statistical learning techniques like Support Vector Machines (SVM), for instance, are employed to classify DNA sequences as HS or non-HS. This paper proposes a method to automate the basic steps in designing an SVM that improves the accuracy of such classification. The method proceeds in two stages and makes use of evolutionary algorithms. An evolutionary algorithm first designs optimal sequence motifs to associate explicit discriminating feature vectors with input DNA sequences. A second evolutionary algorithm then designs SVM kernel functions and parameters that optimally separate the HS and non-HS classes. Results show that this two-stage method significantly improves SVM classification accuracy. The method promises to be generally useful in automating the analysis of biological sequences, and we post its source code on our website.

*Keywords*: DNase I hypersensitive sites; evolutionary algorithms; genetic programming.

## 1. Introduction

Locating DNA noncoding regions that regulate gene transcription is now the remaining challenge in mapping out the entire human genome.[1–3] The latest techniques rely on identifying sites that are hypersensitive to DNA-modifying enzymes and precede regulatory regions, as shown in Fig. 1.[2–8] A wealth of short DNA sequences determined to be hypersensitive (HS) sites are now available from high-throughput experimental techniques.[8,9]

‡Corresponding author.

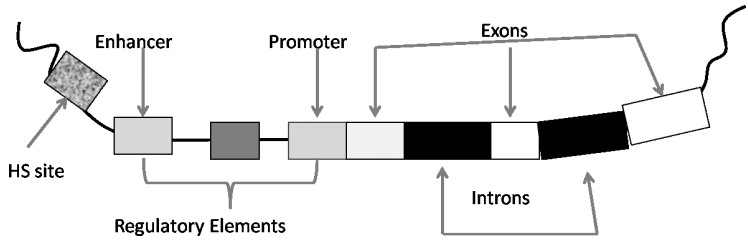



Fig. 1. HS-sites are reliable markers that precede regulatory elements in eukaryotic DNA.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

The abundance of known HS sequences allows applying statistical learning techniques to automate the annotation process. Recent work explores the employment of Support Vector Machines (SVMs) in classifying short DNA sequences as HS or non-HS.[10,11] SVMs have a solid theoretical foundation in statistical learning theory and are the most widely used machine learning technique for binary classification problems.[12] In bioinformatics, SVMs have been applied to predict protein localization sites, DNA translation start sites, and DNA splice sites.[13–18]

Despite their broad applicability, important decisions in constructing an SVM classifier remain problem specific and require understanding the problem domain. Essentially, an SVM maps non-vector data, such as text, graphs, and strings, into a vector space where a hyperplane can be found to optimally separate the two classes of mapped data. The process consists of two basic steps. In the context of classifying input DNA sequences as HS or non-HS, the first step involves associating real-valued feature vectors with the input sequences. The second step involves mapping the feature vectors into a higher-dimensional space where labeled data can be linearly separated by a hyperplane. The success of an SVM classifier depends on both the choice of the feature space and the internal transformation, the *kernel function*. Finding discriminating features that distinguish HS sequences from non-HS sequences is difficult. Even for experts, significant time and resources go to finding features that result in meaningful SVM input vectors.

Choosing a kernel function that transforms the feature vectors into a higher-dimensional space where the data are linearly separable is also problem-specific. In addition, kernel functions have adjustable parameters as does the SVM itself. This generally results in a tedious tuning process with many cycles of experimentation.

In this paper we propose a novel method that removes the need for expert input and automates the two basic components of an SVM classification, feature and kernel selection, all the while improving classification accuracy. Essentially, the method consists of two evolutionary algorithms (EAs). As shown in Fig. 2, the first EA finds a good set of discriminating features.[11] A second EA finds the best kernel with necessary SVM parameters, thus automating the design of a high-performance SVM for sequence classification. Our results show that these two EAs in combination improve SVM classification accuracy in a statistically significant way as compared to other methods.

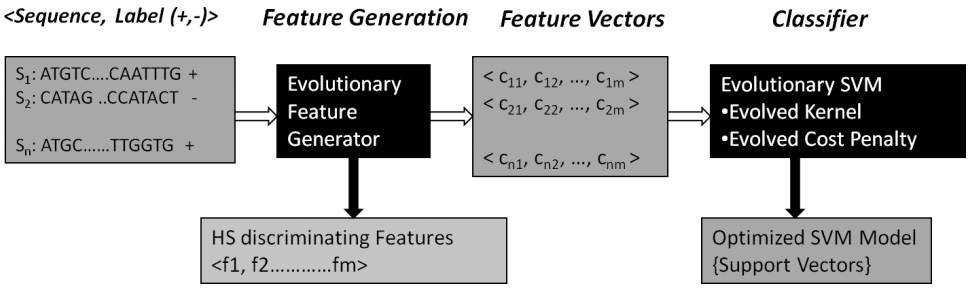



Fig. 2. Overall Algorithm using EAs for both feature and SVM kernel optimization.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

The rest of this paper is organized as follows. A brief summary of related work is provided in Sec. 1.1. Our method is described in Sec. 2. Results follow in Sec. 3. The article concludes with a discussion in Sec. 4.

### 1.1. *Related work*

The issue of extracting meaningful features from biological sequences is circumvented when employing implicit string kernels. These kernels directly associate distances in the feature space through suffix trees or other similarity measures.[19] In other applications, one first associates feature vectors with input sequences and then uses a kernel function to obtain distances in the feature space through dot product calculations.[20] Extracting explicit features has distinct advantages. The features can encapsulate important biological features, and their relative strength or contribution to learning can be directly measured.

Often, the main novelty in applying an SVM to a new classification problem is the extraction of meaningful features that allow converting the input data into vectors. When there is no prior knowledge to guide the design of meaningful features, spectrum features are often employed.[20] A $k$-spectrum is the set of $d = |\Sigma|^k$ features that correspond to all strings of length $k$ ($k$-mers) generated from an alphabet $\Sigma$. A $d$-dimensional feature vector is then associated with an input sequence by recording the frequency of occurrence of each of the $d$ $k$-mers in the sequence. This approach is recently employed to recognize HS sequences.[10]

As the spectrum length increases, the number of features increases exponentially. Various studies reveal that a small percentage of the 6-spectrum features actually contribute to learning.[10,11] Our work in Ref. 11 proposes an EA to explore the space of fixed-length sequences in search of optimal motifs that best discriminate between HS and non-HS sequences. We have shown the benefits of employing these motifs over spectrum features in other bioinformatics applications.[21] We employ the EA proposed in Ref. 11 and briefly summarized in Sec. 2 to obtain features in this work.

The success of an SVM classifier depends on the choice of both the feature space and the kernel function. Many researchers test few predefined kernel functions to

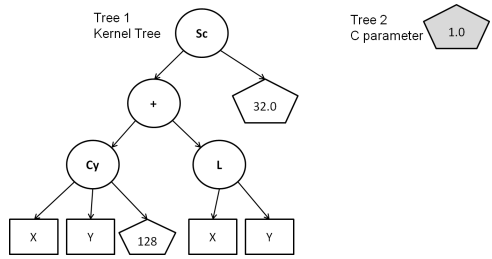



Fig. 3. An individual consisting of an evolved kernel and an evolved cost parameter.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

select one that yields the highest classification accuracy.[22] Grid-based optimization then tunes kernel and SVM parameters.[23] Evolutionary-based algorithms, such as Genetic Programming (GP), are starting to be applied to design an optimal kernel function.[24–27] The kernel functions evolved by the GP in some researches are not guaranteed to follow Mercer's theorem, so optimality is not guaranteed. In all current applications of GP, a small predefined set of kernel functions is employed to evolve new kernel functions.[28,29] The set consists of the Linear, Gaussian, and Polynomial functions, excluding many other known kernel functions. The cost parameter $C$, which controls the trade off between allowing misclassification errors during training of the SVM and forcing rigid margins, is also kept fixed.[28,29]

In this paper, the optimal features obtained with an EA are combined with an optimal kernel function obtained with a GP algorithm. The main novelty of this work is that it simultaneously evolves kernel functions, their parameters, and the SVM cost parameter $C$, while using evolved discriminating features. Figure 3 illustrates the structure of an individual in a GP population. The individual consists of two trees, one containing an evolved kernel that combines Linear, Cauchy and Scalar kernels, and one containing the evolved SVM cost parameter $C$.

## 2. Methods

The EA introduced in our recent work[11] and employed here to obtain meaningful features is summarized below. The section then details the GP algorithm we propose to evolve kernel functions, their parameters, and the SVM cost parameter $C$.

### 2.1. *Finding over-represented motifs in DNA sequences*

The EA we introduce in Ref. 11 searches for motifs that best discriminate between HS and non-HS sequences. The motifs are variable-length strings of length $l$ generated from the the IUPAC code. In addition to the 4-letter $\{A, T, C, G\}$ DNA alphabet, the IUPAC code contains ambiguous symbols that capture groups of nucleotides with shared chemical properties. The motifs vary in length ($l \in \{6, \ldots, 12\}$) because the length of optimal motifs is not known *a priori*. No shorter than 6-mers generated over the DNA alphabet are needed to achieve high SVM classification accuracy.[10]

Our EA uses an island-model approach in which each island contains motifs of the same length (i.e. one-motif species) and evolves in isolation and in parallel with other islands without migration. On each island, a $(\mu + \lambda)$-ES-style EA, where $\mu$ is the number of parents and $\lambda$ is the number of offspring generated. The mutation used is the standard generalization of the bit flip operator to nonbinary alphabets along with a standard 1-point crossover operator.

While the true fitness of an individual should be evaluated in the context of SVM-based classification, doing so on each offspring is computationally impractical. We employ a simpler fitness function that approximates how spectrum features are employed in the kernel function.[10] Given a $k$-mer $w$, the fitness value $f(w) = 100 * |c(w)_{\text{HS}}/\text{total}_{\text{HS}} - c(w)_{\text{non-HS}}/\text{total}_{\text{non-HS}}|$, where $c(w)$ counts the number of sequences containing $w$, and total normalized by the number of known sequences in each class (HS or non-HS). In this way, a motif that is found in all HS sequences but no non-HS sequences (or vice versa), will have the highest fitness score of 100. A motif found with the same frequency in non-HS and HS sequences will have the lowest score of 0. Analysis in our recent work shows that the fitness value of the top motifs strongly correlates with the classification accuracy these motifs confer to an SVM.[11,21] The top 200 motifs of the final GP population are used to construct the feature vectors from the input sequences.

### 2.2. *Our kernel evolution algorithm*

The EA we propose searches over the space of positive semi-definite kernel functions in order to guarantee optimality. We first discuss the concept of *kernel closure* and then detail the elements of the proposed EA.

#### 2.2.1. *Kernel closure*

Kernel functions must be continuous, symmetric, and preferably have a positive (semi)-definite Gram matrix.[30] A positive-definite kernel insures that the optimization problem will be convex and the solution will be unique.[30] New kernel functions can be constructed by combining well-known positive (semi)-definite kernel functions through specific mathematical manipulations that guarantee closure. The new kernel functions inherit the properties of the functions used to construct them.[31] Here we employ an extensive list of known positive (semi)-definite kernel functions (listed in Table 2) to evolve new kernel functions that obey the closure property. The closure-preserving mathematical manipulations we use are listed in Table 1.

#### 2.2.2. *Genetic programming*

Our EA for evolving new kernels is a standard GP algorithm.[32] It simultaneously searches the space of kernel functions (and their parameters) and SVM cost parameters $C$. Each individual in the population is represented as a forest of two trees (see Fig. 4). One tree maintains the evolving kernel function, and the other the

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

Table 1. Operations that guarantee kernel closure.

| Operator | Formula |
|---|---|
| Add | $k(x,y) = k_1(x,y) + k_2(x,y)$ |
| Scalar | $k(x,y) = a \cdot k_1(x,y)$ |
| Multiply | $k(x,y) = k_1(x,y) \cdot k_2(x,y)$ |
| Exponential | $k(x,y) = e^{k_1(x,y)}$ |

Table 2. All terminal and nonterminal nodes are shown.

| Name | Formula | Args, constrs |
|---|---|---|
| Polynomial (P) | $k(x,y) = (\alpha x^T y + c)^d$ | 3, $c$, $d \in \mathcal{R}$ |
| Linear (L) | $k(x,y) = x^T y + c$ | 3, $c \in \mathcal{R}$ |
| Sigmoid (S) | $k(x,y) = e^{-\sigma \|x-y\|^2}$ | 3, $\sigma \in \mathcal{R}$ |
| Laplace (Lp) | $k(x,y) = e^{-\frac{\|x-y\|}{\sigma}}$ | 3, $\sigma \in \mathcal{R}$ |
| Anova (A) | $k(x,y) = (\sum_{k=1}^{n} e^{-\sigma(x_k - y_k)^2})^d$ | 3, $\sigma \in \mathcal{R}$ |
| Rational Quadratic (RQ) | $k(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2 + c}$ | 3, $c \in \mathcal{R}$ |
| Inv. MultiQuadratic (IQ) | $k(x,y) = \frac{1}{\sqrt{\|x-y\|^2 + c^2}}$ | 3, $c \in \mathcal{R}$ |
| Circular (CLR) | $k(x,y) = \frac{2}{\pi} \arccos\left(-\frac{\|x-y\|}{\sigma}\right) - \frac{2}{\pi}\frac{\|x-y\|}{\sigma}\sqrt{1 - \frac{\|x-y\|^2}{\sigma}}$ | 3, $\sigma \in \mathcal{R}$ |
| Spherical (SPL) | $k(x,y) = 1 - \frac{3}{2}\frac{\|x-y\|}{\sigma} + \frac{1}{2}\left(\frac{\|x-y\|}{\sigma}\right)^3$ if $\|x-y\| < \sigma$ | 3, $\sigma \in \mathcal{R}$ |
| Wave (W) | $k(x,y) = \frac{\theta}{\|x-y\|} \sin\left(\frac{\|x-y\|}{\theta}\right)$ | 3, $0 \leq \theta < 2\pi$ |
| Spline (SLN) | $k(x,y) = \Pi_{i=1}^{d}\, 1 + x_i y_i + x_i y_i \min(x_i, y_i) - \frac{x_i + y_i}{2}\min(x_i, y_i)^2 + \frac{\min(x_i, y_i)^3}{3}$ | 3, $d \in \mathcal{I}$ |
| Bessel (B) | $k(x,y) = \frac{J_{v+1}(\sigma\|x-y\|)}{\|x-y\|^{-n(v+1)}}$ | 4, $n \in \mathcal{I}$ |
| Cauchy (Cy) | $k(x,y) = \frac{1}{1 + \frac{\|x-y\|^2}{\sigma}}$ | 3, $\sigma \in \mathcal{R}$ |
| Chi-Square (CHI) | $k(x,y) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$ | 2 |
| Histogram (HI) | $k(x,y) = \sum_{i=1}^{n} \min(x_i, y_i)$ | 2 |
| T-Student (T-s) | $k(x,y) = \frac{1}{1 + \|x-y\|^d}$ | 3, $\sigma \in \mathcal{R}$ |
| Add (+) | $k(x,y) = k_1(x,y) + k_2(x,y)$ | 2 Kernels |
| Multiply ($*$) | $k(x,y) = k_1(x,y) \cdot k_2(x,y)$ | 2 Kernels |
| Scalar (Sc) | $k(x,y) = a \cdot k_1(x,y)$ | 1 Kernel, $a \in \mathcal{R}$ |
| Exponential (E) | $k(x,y) = e^{k_1(x,y)}$ | 1 Kernel |
| $x_i, \ldots, x_n$ | | 0, input |
| $y_i, \ldots, y_n$ | | 0, input |
| ERC-int | | Integer |
| ERC | | Real |

cost parameter $C$. The tree that maintains $C$ consists of only one node, and is subjected only to the mutation operator. The representation of the kernel tree in each individual is analogous to the parse trees employed for Lisp expressions. Each nonterminal node in a kernel tree is either a mathematical operator (add, scalar, multiply, exponential) or a predefined kernel function. Each terminal node in a kernel tree is either a kernel parameter (e.g. $\sigma$, $d$, $\theta$) or one the input feature vectors $x$, $y$. Table 2 provides a complete lists of the terminal and nonterminal nodes.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

The mutation operator is applied to both parameters and kernel functions. A tree is first picked at random (out of the tree maintaining the kernel function and the single-node tree maintaining the cost parameter $C$). A node is than picked at random from the selected tree. Every parameter located in the subtree rooted at the selected node is then mutated according to a Gaussian probability distribution over its preset range. When the selected tree is the kernel tree, mutation is implemented through the growth method.[32] A node picked at random from the kernel tree is replaced by a randomly-generated subtree, as illustrated in Fig. 4.

The crossover operator applies only to the kernel tree. Parents are selected using standard tournament selection, and the standard Koza-style crossover mechanism is employed to generate an offspring kernel function. Compatible parents are sought; that is, individuals are randomly sampled from a population until two are found whose kernel trees have the same constraints. A random node is then chosen in each parent tree such that the two nodes have the same return type. If, by swapping subtrees at these nodes, the two trees do not violate maximum depth constraints, the swap is performed. Otherwise, the hunt for random nodes is repeated. Figure 4 provides an illustration of crossover.

Following the lead of Ref. 33, we start with a large initial population (in this paper, 2,000) and then systematically reduce it by 20% in each generation. This

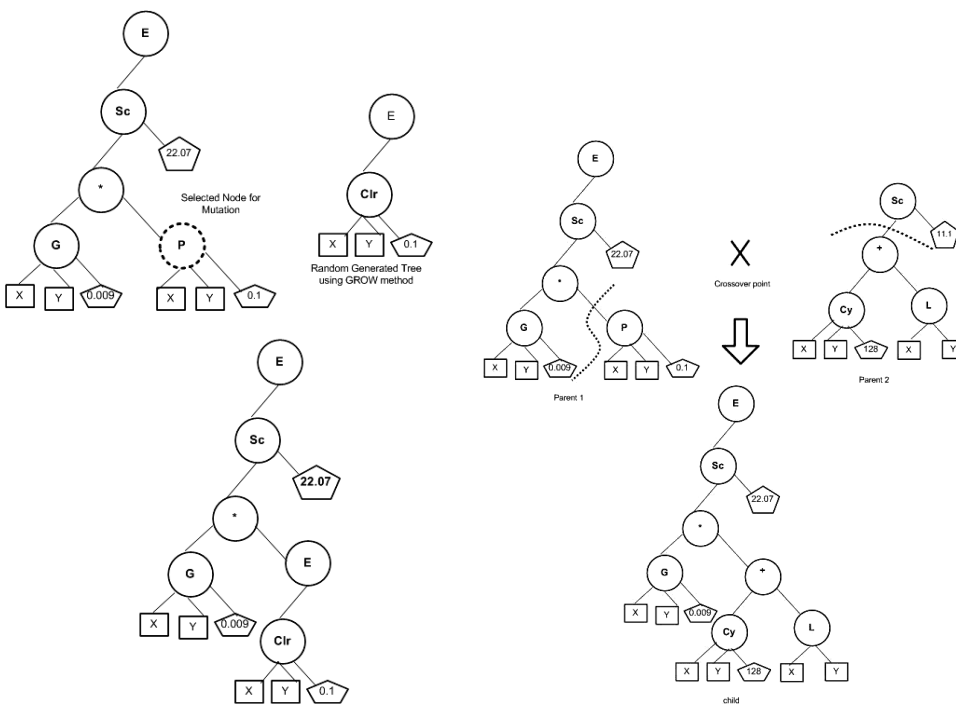



Fig. 4. Mutation (left) and crossover (right) are illustrated on sample kernel trees.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

generally makes the best use of a fixed computation budget in GP. Similarly, we use the standard lexicographic parsimony pressure to help control bloat.[34]

#### 2.2.3. *Putting it all together*

The optimal features are evolved separately through the EA proposed in our previous work and employed here to convert input sequences into feature vectors.[11] The fitness of each individual in the population of our GP algorithm is measured as the classification accuracy obtained when applying an SVM with the kernel function and cost parameter $C$ in the individual on the feature vectors.

Input sequences are obtained from noble.gs.washington.edu/proj/hs. They consist of 280 HS sequences and 737 non-HS sequences experimentally obtained from throughout the human genome. HS and non-HS sequences have similar average lengths of 242 nucleotides.

The performance of the SVM is tested via 10-fold cross-validation on the total set of 1,017 HS and non-HS sequences. The set is randomly divided into 10 subsets of equal size. The SVM is trained on 90% of the subsets and tested on the 10% held out. This is referred to as 10-fold validation. The area under the receiver operating characteristic (ROC) curve is reported as an average over the 10-fold validations.

The method is implemented in Java and run on an Intel Core2 Duo machine with 4GB RAM and 2.66GHz CPU. The EA implementation builds upon the GP implementation of ECJ software.[35] We use the open-source Biojava project[36] to integrate bioinformatics utilities for genomic sequences and the libSVM package[37] for the SVM implementation. The source code of the method and various development details are publicly available at http://cs.gmu.edu/∼ashehu/?q=KernelOptimization.

## 3. Results

The EA is run 30 different times to obtain 30 different sets of top motifs which become features in SVM. In each case, the resulting feature vectors are tested with the SVM cross-validation test, employing a default kernel and default parameters. The kernel used is the radial basis function (RBF) shown to outperform other predefined kernels in our recent work.[11] The average accuracy obtained with this kernel over 30 different runs, each of which results in a different set of 200 top-scoring motifs, is 82.9, with a standard deviation 1.1. The maximum and minimum accuracies obtained over these runs are 77.1 and 85.15, respectively.

Three sets of feature vectors, the best, average, and worst, features, from 30 runs were then run with SVM tuned with the grid search technique[37] that tunes the cost parameter C and the RBF kernel parameter $\gamma$. The same three sets of features were individually run with evolutionary algorithm, evolving the kernels and the cost parameter. Since the evolutionary approach is stochastic, we froze each of the above feature sets and ran the kernel evolution 30 times on each set.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

Table 3. Accuracies obtained with default kernel, the tuned RBF kernel and evolved kernel when employing best, average and worst features.

| Feature set | Default kernel | Grid tuned kernel | Evolved kernel $\mu$,$\sigma$ |
|---|---|---|---|
| Best feature set | 85.15 | 85.39 | 87.3, 0.1 |
| Average feature set | 82.9 | 83.783 | 85.9, 0.1 |
| Worst feature set | 77.1 | 79.3 | 81.57, 0.4 |

Table 4. Fitness values are shown for some of the top individuals (kernel function and cost parameter) obtained with our GP algorithm for Best, Average and Worst Feature Sets.

| Set | Evolved kernel | C | Accuracy |
|---|---|---|---|
| Best | $(*(CY(x)(y)(6.1394E-4))(S(x)(y)(7.4104004E-4)))$ | 17.85 | 87.35 |
| Best | $(CLR(x)(y)(10.746466))$ | 1.18 | 86.85 |
| Best | $(CY(x)(y)(2.3240509E-4))$ | 1.68 | 86.81 |
| Best | $(Sc(SPL(x)(y)(56.92857))(48.207794))$ | 1.00 | 86.79 |
| Best | $(*(CY(x)(y)(30.50425))(SPL(x)(y)(92.285255)))$ | 1.24 | 86.77 |
| Average | $(E(E(CLR(x)(y)(21.81946))))$ | 0.1297 | 86.19 |
| Average | $(Sc(E(CLR(x)(y)(10.341912)))(27.86481))$ | 11.21 | 85.97 |
| Average | $(E(*(CLR(x)(y)(18.004602))(CLR(x)(y)(20.13649))))$ | 1.23 | 85.78 |
| Average | $(*(S(x)(y)(6.9958554E-4))(Linear(x)(y)))$ | 14.1 | 85.74 |
| Average | $(CLR(x)(y)(13.197769))$ | 1.1 | 85.51 |
| Worst | $(G(x)(y)(8.8855857E-4))$ | 19.18246 | 82.64 |
| Worst | $((T-s(x)(y)(26.574564))$ | 23.206886 | 81.56 |
| Worst | $(Sc(S(x)(y)(5.8920565E-4))(2.1301475))$ | 41.87304 | 81.41 |
| Worst | $(E(G(x)(y)(2.7014833E-4)))$ | 32.856094 | 81.39 |
| Worst | $(*(S(x)(y)(6.7382515E-4))(E(CLR(x)(y)(28.763956))))$ | 22.6613 | 81.31 |

Table 3 shows the comparative analysis of accuracies of default RBF kernel, grid search tuned kernel and evolved kernel.

Table 4 illustrates the top five individuals (kernel and cost parameters $C$) that were evolved for each of the three feature sets (Best, Average, and Worst).

To compare the best features and top kernel evolved in SVM with our hybrid methodology for comparative analysis, we took the same top best features which gave us 87.35 accuracy with the top evolved kernel, as input for other classifiers. The classifiers chosen are based on previous research in sequence classifications which had yielded better results.[38] Table 5 summarizes the comparative analysis of our approach with various traditional classification systems. The Appendix defines the parameters for each of the methods.

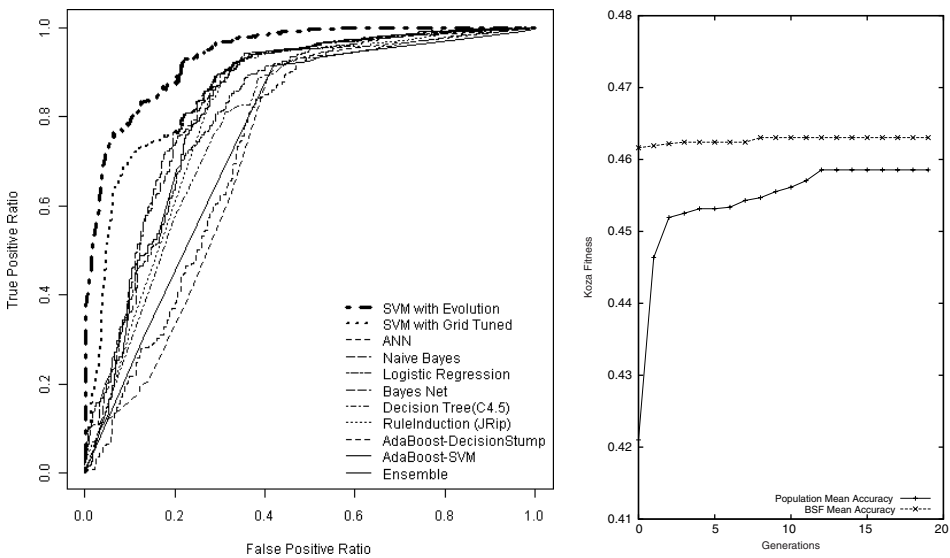
The ROC curves for all the classifiers in comparison with our evolutionary algorithm are also plotted in Fig. 5(a) to show statistically significant difference in ROC area comparisons. Immediate convergence is seen in the Evolutionary runs, i.e. population mean and best-so-far mean converge within 20 generations. The plot in Fig. 5(b) shows the same for Best Feature set to illustrate the convergence and validate our design decision of having ever-reducing population.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

Table 5. Ten-Fold Cross Validation accuracy, Precision, Recall and Root Mean Square Error (RMSE) Comparison with various classifiers using the Best Features from Feature Evolution.

| Classification method | CV-Accuracy | Precision | Recall | RMSE |
|---|---|---|---|---|
| SVM (Kernel Evolution) | 87.35 | 87.6 | 87.8 | 0.343 |
| SVM (Grid Tuned RBF Kernel) | 85.39 | 85.56 | 85.58 | 0.371 |
| Bayesian | 84.56 | 84.3 | 84.6 | 0.389 |
| Bayesian Net | 83.67 | 83.8 | 83.7 | 0.3979 |
| Logistic Regression | 82.3009 | 81.6 | 82.3 | 0.3682 |
| Neural Network | 79.8427 | 79.1 | 79.8 | 0.4241 |
| Decision Tree (C4.5) | 82.0059 | 81.2 | 82.0 | 0.4035 |
| Rule Induction (Ripper) | 82.2026 | 81.5 | 82.2 | 0.3824 |
| AdaBoost (Decision Stumps) | 81.9076 | 81.1 | 81.9 | 0.3814 |
| AdaBoost (SVM) | 84.6608 | 84.2 | 84.7 | 0.3581 |
| Ensemble Classifier (SVM,C4.5,logistic) | 84.1691 | 83.8 | 84.2 | 0.3665 |

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.




(a) (b)

Fig. 5. (a) ROC curve comparison with different classifiers (a) and (b) fitness versus generation.

## 4. Conclusion

Our results show that employing an EA algorithm to obtain meaningful features and a GP algorithm to obtain new better kernel functions and values for the SVM cost parameter $C$ significantly improve the accuracy of an SVM classification for the HS recognition problem. Moreover, the employment of evolutionary computing automates important SVM design decisions, reducing the dependency on *a priori* expert knowledge and time-consuming experimentation.

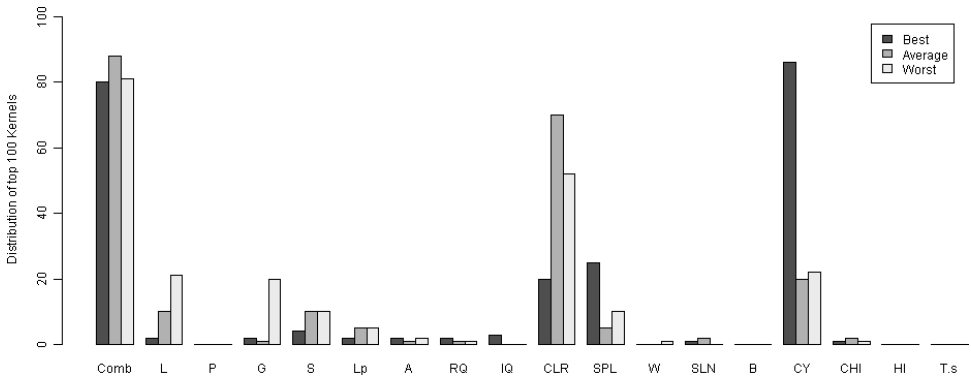



Fig. 6. Distribution of the top 100 kernels in the final population for Best, Average, and Worst feature sets.

Some of the top kernels obtained by our GP algorithm are a combination of the exponential, multiplication, or scalar operators over the Gaussian, Sigmoid, and Linear kernels. The presence of the Cauchy, Circular, Spiral, and Student-T kernels among top-scoring ones further justifies our employment of an extensive list of kernel functions in our GP algorithm. Figure 6 shows in detail the distribution of kernel functions found among the top 100 kernels evolved by our GP algorithm. Results are shown for each Best, Average, and Worst feature sets. The largest percentage consists of combined kernels evolved by our GP algorithm. Each of the predefined kernel functions can be found at lower percentages in this distribution.

We emphasize that our GP algorithm maintains closure and guarantees the optimality of obtained kernels. As a consequence, independent of the feature set used, the evolved kernels show an improvement in classification accuracy. Our hybrid approach of combining that with evolved features yields additional improvement. The result is an SVM that produces statistically significantly better classification performance than most well known documented classification methods.

While the fitness of an evolved kernel function is estimated in the context of the SVM classification, the fitness of the motifs sought to associate meaningful feature vectors with input sequences employs a simpler filter function in the interest of keeping computational cost low. Since the fitness of a kernel function is intimately dependent on the feature vectors employed in the SVM classification, our future work will consider integrated evolutionary schemes that evolve kernels and motifs in tandem. Co-evolution of features and kernels may further boost classification accuracies but at an expected computational cost. Distributed implementations will be sought to manage the computational cost.

## Acknowledgments

We are indebted to Sean Luke and Keith Sullivan for insights on GP and Rezarta Dogan and Sarang Kayande for discussions on this work.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

## Appendix A. Classifier Implementation Details

We will list machine learning classifier as well as list various parameters in evolutionary algorithm used by our methodology. Most of the Classifiers were parameter tuned by making sure various options suited for these methods are researched, tried and the best one with changes to default settings are given below.

Table A. Classifier parameters used in our comparative study.

| Classifier type | Parameter changed |
|---|---|
| Naive Bayes | useSupervisedDiscretization = true |
| Bayesian Net | estimator = BMAEstimator |
| Logistic Regression | ridgeValue = 1E−12 |
| Neural Network | LearningRate = 0.1, epoch = 700 |
| Decision Tree(C4.5) | useLaplace = true |
| Rule Induction(JRip) | folds = 10, optimizations = 3 |
| AdaBoostM1 | useResampling = true |
| Ensemble Classifier | algorithm = Forward + Backward, replacement = true |

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

## Appendix B. Evolutionary Algorithm Details

### B.1. *Feature evolution algorithm*

Population Size = 5000, Generations = 50, Parent Selection = Uniform Stochastic, Survival Selection = Truncation, Crossover = one_point, Mutation = bit flip, and probability of mutation = 1/genome_length.

### B.2. *Kernel evolution algorithm*

Population Size = 2000, Generations = 30, Initialization = Half and Half method, Crossover = Subtree with probability 0.8, Mutation = Grow with probability of 0.2, Elites = 10. The entire implementation is available as open source with samples is available http://cs.gmu. edu/∼ashehu/?q=KernelOptimization.

## References

1. Consortium IHGS, Finishing the euchromatic sequence of the human genome, *Nature* **431**(7011):931–945, 2004.
2. Maston GA, Evans SK, Green MR, Transriptional regulatory elements in the human genome, *Annu Rev Genom Human Genet* **7**:29–59, 2006.
3. Higgs DR, Vernimmen D, Hughes J, Gibbons R, Using genomics to study how chromatin influences gene expression, *Annu Rev Genom Human Genet* **8**:299–325, 2007.
4. Wu C. The 5′ ends of Drosophila heat shock genes in chromatin are hypersensitive to DNase I, *Nature* **286**(5776):854–860, 1980.
5. Gross DS, Garrard WT, Nuclear hypersensitive sites in chromatin, *Annu Rev Biochem* **57**:159–197, 1988.
6. Lowrey CH, Bodine DM, Nienhuis AW, Mechanism of DNase I hypersensitive site formation within the human globin locus control region, *Proc Natl Acad Sci USA* **89**(3):1143–1147, 1992.

7. Burgess-Beusse B, Farrell C, Gaszner M, Litt M, Mutskov V, Recillas F-Targa, Simpson M, West A, Felsenfeld G, The insulation of genes from external enhancers and silencing chromatin, *Proc Natl Acad Sci USA* **99**(S4):16433–16437, 2002.
8. Sabo PJ, Humbert R, Hawrylycz M, Wallace JC, Dorschner MO, McArthur M, Stamatoyannopoulos JA, Genome-wide identification of DNase I hypersensitive sites using active chromatin sequence libraries, *Proc Natl Acad Sci USA* **101**(13):4537–4542, 2004.
9. Dorschner MO, Hawrylycz M, Humbert R, Wallace JC, Shafer A, Kawamoto J, Mack J, Hall R, Goldy J, Sabo PJ, Kohli A, Li Q, McArthur M, Stamatoyannopoulos JA, High-throughput localization of functional elements by quantitative chromatin profiling, *Nat Methods* **1**(3):219–225, 2004.
10. Noble WS, Kuehn S, Thurman R, Yu M, Stamatoyannopoulos JA, Predicting the *in vivo* signature of human gene regulatory sequences, *Bioinformatics* **21**(1):i338–i343, 2005.
11. Kamath U, De Jong KA, Shehu A, Selecting predictive features for recognition of hypersensitive sites of regulatory genomic sequences with an evolutionary algorithm, *GECCO: Gen Evol Comp Conf*, pp. 179–186, New York, NY, USA, 2010. ACM.
12. Vapnik VN, *Statistical Learning Theory*, Wiley & Sons, New York, NY, 1998.
13. Habib T, Zhang C, Yang JY, Yang MQ, Deng Y, Supervised learning method for the prediction of subcellular localization of proteins using amino acid and amino acid pair composition, *BMC Genom* **9**(1):S1–S16, 2008.
14. Zien A, Raetsch G, Mika S, Schölkopf B, Lengauer T, Mueller KR, Engineering support vector machine kernels that recognize translation initiation sites, *Bioinformatics* **16**(9):799–807, 2000.
15. Zhang XH, Heller KA, Hefter I, Leslie CS, Chasin LA, Sequence information for the splicing of human pre-mrna identified by support vector machine classification, *Genome Res* **13**(12):2637–2650, 2003.
16. Islamaj-Dogan R, Getoor L, Wilbur WJ, Mount SM, Features generated for computational splice-site prediction correspond to functional elements, *BMC Bioinformatics* **8**:410–416, 2007.
17. Islamaj-Dogan R, Getoor L, Wilbur WJ, A feature generation algorithm with applications to biological sequence classification, in Liu H, Motoda H (eds.), *Computational Methods of Feature Selection*, Springer, Berlin, Heidelberg, 2007.
18. Noble WS, Support vector machine applications in computational biology, in Schölkopf B, Tsuda K, Vert J-P (eds.), *Kernel Methods in Computational Biology*, MIT Press, Cambridge, MA, 2004.
19. Leslie C, Kuang R, Bennett K, Fast string kernels using inexact matching for protein sequences, *J Mach Learn Res* **5**:1435–1455, 2004.
20. Noble WS Leslie CS, Eskin E, The spectrum kernel: A string kernel for SVM protein classification, *Pacific Symposium on Biocomputing*, Vol. 7, pp. 564–575, Baoding, China, 2002.
21. Kamath U, Shehu A, De Jong KA, Using evolutionary computation to improve svm classification, *WCCI: IEEE World Conf Comp Intel* IEEE Press, 2010, in press.
22. Boser BE, Guyon IM, Vapnik VN, A training algorithm for optimal margin classifiers, in Haussler D (ed.), *5th Annual ACM Workshop on COLT*, pp. 144–152. ACM Press, 1992.
23. Staelin C, Parameter selection for Support Vector Machines, 2002.
24. de Souza BF, de Carvalho AC, Calvo R, Ishii RP, Multiclass SVM model selection using particle swarm optimization, *Sixth Int Conf Hybrid Intelligent Systems*, 2006.


J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

25. Huang C-L, Wang C-J, A ga-based feature selection and parameter optimization for support vector machines, *Expert Systems with Applications*, pp. 231–240, 2006.
26. Phienthrakul T, Kijsirikul B, Evolutionary strategies for multi-scale radial basis function kernels in support vector machines, *Genetic and Evolutionary Computation Conference*, pp. 905–911, Washington D.C, USA, 2005.
27. Friedrichs F, Igel C, Evolutionary tuning of multiple svm parameters, *12th European Symposium on Artificial Neural Networks (ESANN 2004*, pp. 519–524, 2004.
28. Gagné C, Schoenauer M, Sebag M, Tomassini M, Genetic programming for kernel-based learning with co-evolving subsets selection, in *Parallel Problem Solving From Nature, Reykjavik, LNCS*, pp. 1008–1017, Springer Verlag, Berlin, 2006.
29. Sullivan K, Luke S, Evolving kernels for support vector machine classification, *Genetic and Evolutionary Computation Conference*, 2007.
30. Schölkopf B, Smola AJ, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Boston, USA, 2002.
31. Shawe-Taylor J, Cristianini N, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004.
32. Koza J, *On the Programming of Computers by Means of Natural Selection*, MIT Press, Boston, MA, 1992.
33. Luke S, Balan GC, Panait L, Population implosion in genetic programming, in *Genetic and Evolutionary Computation Conference*, 2003.
34. Mierswa I, Evolutionary learning with kernels, a generic solution for large margin problems, in *GECCO: Gen Evol Comp Conf* pp. 1553–1560, 2006.
35. Luke S, Panait L, Balan G, Paus S, Skolicki Z, Popovici E *et al.*, ECJ: A Java-Based Evolutionary Computation Research, 2010.
36. Holland RC, Down TA, Pocock M, Prlic A, Huen D, James K *et al.*, BioJava: An open-source framework for bioinformatics, *Bioinformatics* **24**(18):2096–2097, 2008.
37. Fan R-E, Chen P-H, Lin C-J, Working set selection using the second order information for training SVM, *J Mach Learn Res* **6**(1532–4435):1889–1918, 2005.
38. Tavares LG, Lopes HS, Erig Lima CR, A comparative study of machine learning methods for detecting promoters in bacterial DNA sequences, *ICIC (2)*, pp. 959–966, 2008.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.

**Uday Kamath** received his B.S. in Electrical Electronics from Bombay University in 1996 and Masters in Computer Science from University of North Carolina at Charlotte in 1999. He is currently pursuing his Ph.D. in Computer Science at George Mason University. He also works as Technical Architect in Analytics and Detection group of Norkom Technologies, concentrating on using machine learning, evolutionary software and statistical modeling techniques in various fraud detection domains. His research interest includes the applications of evolutionary computation methods to finance and to computational biology and bioinformatics. He is a member of the IEEE and the ACM.

**Amarda Shehu** is an Assistant Professor in the Department of Computer Science at George Mason University. She holds affiliated appointments in the Department of Bioinformatics and Computational Biology and the Bioengineering Program at George Mason University. She earned her Ph.D. in Computer Science at Rice University in Houston, TX, in 2008, where she was also an NIH fellow of the Nanobiology Training Program of the Gulf Coast Consortia. Her research encompasses probabilistic search frameworks, evolutionary algorithms, and machine learning for problems in computational biology and biophysics. Dr. Shehu is a member of the IEEE and ACM.

**Kenneth A. De Jong** is a Professor of Computer Science and Associate Director of the Krasnow Institute at George Mason University. Dr. De Jong's research interests include evolutionary computation, adaptive systems and machine learning. He is an active member of the Evolutionary Computation research community with a variety of papers and presentations in this area. He is also responsible for many of the workshops and conferences on Evolutionary Algorithms. He is the founding editor-in-chief of the *Journal Evolutionary Computation* (MIT Press), and a member of the board of ACM SIGEVO. He is the recipient of an IEEE Pioneer award in the field of Evolutionary Computation and a lifetime achievement award from the Evolutionary Programming Society. Dr. De Jong is a member of IEEE and ACM.

J. Bioinform. Comput. Biol. 2011.09:399-413. Downloaded from www.worldscientific.com
by GEORGE MASON UNIVERSITY on 07/04/13. For personal use only.