Feature and Kernel Evolution for Recognition of Hypersensitive Sites in DNA Sequences

Uday Kamath[1], Amarda Shehu[1,2], and Kenneth A. De Jong[1]

[1] Department of Computer Science
[2] Department of Bioinformatics and Computational Biology
George Mason University, Fairfax, VA, 22030
kdejong@gmu.edu

Abstract. The annotation of DNA regions that regulate gene transcription is the first step towards understanding phenotypical differences among cells and many diseases. Hypersensitive (HS) sites are reliable markers of regulatory regions. Mapping HS sites is the focus of many statistical learning techniques that employ Support Vector Machines (SVM) to classify a DNA sequence as HS or non-HS. The contribution of this paper is a novel methodology inspired by biological evolution to automate the basic steps in SVM and improve classification accuracy. First, an evolutionary algorithm designs optimal sequence motifs used to associate feature vectors with the input sequences. Second, a genetic programming algorithm designs optimal kernel functions that map the feature vectors into a high-dimensional space where the vectors can be optimally separated into the HS and non-HS classes. Results show that the employment of evolutionary computation techniques improves classification accuracy and promises to automate the analysis of biological sequences.

Keywords: DNase I hypersensitive sites, evolutionary algorithms, support vector machines, genetic programming, kernel functions, motifs.

1 Introduction

Many diseases and phenotypical differences among cells are caused by variations in non-coding regions of DNA that regulate gene transcription [29,15]. Since the successful annotation of the human genome with functional coding regions [6], locating regulatory regions is now the remaining challenge to mapping out the entire human genome [29,15]. Based on the observation that regulatory regions bind with transcription-factor binding proteins to activate or repress following genes, many experimental techniques originally relied on detecting transcription-factor binding sites to locate regulatory regions. This approach has proved costly and time consuming [1]. Current techniques rely instead on identifying sites that precede regulatory regions and are particularly sensitive to DNA-modifying enzymes like non-specific endonuclease DNase I [41,13,26,4,35,29,15]. A wealth of short DNA sequences determined to be hypersensitive (HS) sites are now available from high-throughput experimental techniques [35,9].

J. Suzuki and T. Nakano (Eds.): BIONETICS 2010, LNICST 87, pp. 213–228, 2012.

Despite the discovery that HS sites are reliable markers of regulatory regions, rapid annotation of the entire human genome requires a combination of experimental and computational techniques [33]. The abundance of HS sequences already identified in wet laboratories allows applying statistical learning techniques to automate the annotation process. Recent work explores the employment of Support Vector Machines (SVM) to the binary classification problem of classifying short DNA sequences as HS or non-HS [33,21]. SVMs have a solid theoretical foundation in statistical learning theory and are the most widely used machine learning technique in binary classification problems [39]. In bioinformatics, SVMs have been applied to predict protein localization sites [14], DNA translation start sites [43], DNA splice sites [42,20,19], and more (cf. to [32]).

Despite their broad applicability, important decisions in an SVM classifier remain problem specific and require some understanding of 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 vectors into the two available classes. 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 feature vectors with the input sequences. The second step involves mapping the feature vectors into a high-dimensional space where labeled data can be linearly separable by a hyperplane. Once the hyperplane is computed, predicting the label of an unlabeled data point involves determining on which side of the hyperplane the point lies.

The success of an SVM classifier depends on both the choice of the feature space and the internal transformation, the *kernel function*, used. 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. For instance, if training sequences belonging to one class are known to contain specific subsequences with higher frequency than the sequences belonging to the other class, these subsequences could be used as features, and their frequency of occurrence can be used to convert an input sequence into a feature vector. Such information is not available to non-experts, and significant time and resources are often devoted to finding features that give meaningful vectors.

The particular choice of a kernel function that transforms the feature vectors into a high-dimensional space where the data are linearly separable is also problem-specific. Finding an optimal kernel function is nontrivial, and many researchers rely on testing a small number of predefined kernels. Well-known kernel functions include the Linear, Polynomial, Radial Basis, Gaussian Radial, and Sigmoid kernel functions [2]. One needs to determine not only the kernel function that yields the highest classification accuracy, but also the optimal values for the various parameters contained in the kernel function. Finding the right kernel function and the right parameters for the selected kernel function is a tedious optimization process requiring many cycles of experimentation.

The contribution of this paper is a novel methodology 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.

The methodology is inspired by biological evolution and employs evolutionary computation techniques to evolve optimal features and optimal kernel functions. We have recently proposed an evolutionary algorithm (EA) to design optimal features [21]. The novelty in the work presented here is the combination of these features with novel optimal kernel functions evolved through a novel genetic programming (GP) algorithm. Our results show that the employment of these evolutionary computation techniques to select optimal features and optimal kernel functions automates the process of SVM classification and significantly improves the classification accuracy.

The rest of this paper is organized as follows. A brief summary of related work is provided in section 1.1. Our method is described in section 2. Results follow in section 3. The article concludes with a discussion in section 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 [24]. 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 [25]. 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 no prior knowledge is available to guide the design of meaningful features, spectrum features are often employed to explicitly map an input sequence to a vector space [25]. 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 Σ. 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. Such an approach is employed in [33] to to recognize HS sequences.

Using spectrum features, however, has the disadvantage of an exponential increase in the number of features as the spectrum length increases. For instance, the 6-mers employed in [33] result in 4096 features. A high number of features adversely impacts the performance of the SVM, both in terms of running time and classification accuracy. Analysis in [33] and our recent work [21] reveals that a very small percentage of the 6-spectrum features actually contribute to learning. These observations, together with our hypothesis that sequence motifs in HS sequences may make for better features, motivated our recent work [21].

Instead of enumerating fixed-length sequences, our work in [21] 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. Employing these motifs instead of spectrum features improves the classification accuracy by as much as 10% [21]. We have shown the benefits of employing these motifs over spectrum

features in other bioinformatics applications [22]. We employ the EA proposed in [21] and briefly summarized in section 2 to obtain features in this work.

The success and effectiveness of an SVM classifier depends not only on the choice of the feature space, but also the selected kernel function. Many researchers test a small number of predefined kernels functions, such as Linear, Polynomial, Radial Basis, Gaussian Radial, and Sigmoid [2], to select one that yields the highest classification accuracy. Many of the kernel functions contain parameters that need further tuning to improve accuracy. Finding the right kernel function and the right parameters can be a tedious optimization process.

A heuristics-based grid search technique is often employed to tune kernel parameters [28]. Particle Swarm Optimization and Genetic Algorithms (GAs) have been employed to find optimal parameters in a Gaussian kernel in [8,18]. Evolutionary-based methods are beginning to be applied not only to find optimal parameters in a selected kernel function, but also to design an optimal kernel function [34,11]. GP is employed in [12] to evolve kernel functions. The functions in [12] are not guaranteed to follow Mercer's theorem, so optimality is not guaranteed. Additionally, in all current applications of GP, a small predefined set of kernel functions is employed to evolve new kernel functions [12,38]. The set includes only the Linear, Gaussian, and Polynomial functions, excluding many other known kernel functions. Additionally, the cost parameter C, which controls the trade off between allowing misclassification errors during training of the SVM and forcing rigid margins, is kept at a fixed value in [12,38].

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 the work presented here is a novel approach that allows simultaneously evolving kernel functions, their parameters, and the SVM cost parameter C. Unlike the existing work summarized above, an extensive list of available kernel functions is employed to evolve new functions. Additionally, the evolution of the kernel functions in our GP is subjected to the Mercer's theorem that kernel functions be positive semi-definite [36], thus guaranteeing optimality.

2 Methods

The EA introduced in our recent work [21] and employed here to obtain meaningful features is briefly summarized below. The rest of the section details the novel 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 [21] essentially searches for motifs that best discriminate between HS and non-HS sequences. The motifs are variable-length strings of length $l \in \{6, \ldots, 12\}$ generated from the the IUPAC code [5] shown in Table 1. In addition to the 4-letter $\{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$ DNA alphabet, the IUPAC code contains ambiguous symbols that allow specifying groups of nucleotides with

shared chemical properties. The motifs vary in length from 6-mers to 12-mers because no a priori information is available on the length of optimal motifs. Additionally, work in [33], which employs 6-mers generated over the 4-letter alphabet of DNA, shows that no shorter than 6-mers are needed to achieve high classification accuracy with an SVM.

Table 1. IUPAC code is adapted from [5]

Symbol	Meaning	Description Origin
G	G	**G**uanine
A	A	**A**denine
T	T	**Thymine**
C	C	**C**ytosine
R	G or A	pu**R**ine
Y	T or C	p**Y**rimidine
M	A or C	a**M**ino
K	G or T	**K**etone
S	G or C	**S**trong interaction
W	A or T	**W**eak interaction
H	A or C or T	**H** follows G in alphabet
B	G or T or C	**B** follows A in alphabet
V	G or C or A	**V** follows U in alphabet
D	G or A or T	**D** follows C in alphabet
N	G or A or T or C	a**N**y

The EA searches the space of candidate motifs using a $(\mu + \lambda)$-ES-style EA, where μ is the number of parents and λ is the number of offsprings generated as each population of motifs evolves. The first population contains μ randomly generated motifs. In each generation, parents are selected uniformly at random to produce λ offsprings through mutation and crossover. Truncation is employed to determine which of the μ fittest individuals (motifs) will survive as the next generation of parents. In our recent work [21] and experiments here, $\mu = 500$ and $\lambda = 200$. Our recent work [21] additionally shows that the island-model approach yields better motifs than crossbreeding motifs of different lengths, which is confirmed by other work [40,7]. In the island-model approach, 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. It is also interesting to point out that both in nature and evolutionary algorithms, offsprings produced by structurally dissimilar parents are inviable.

Given a current population of motifs, there is an equal chance of applying either mutation or crossover to generate a new offspring. If the mutation operator is chosen, each motif in the current population has equal probability of being selected as a parent. The mutation operator is equivalent to the bit flip operator. Any of the symbols of the chosen parent has equal probability of being mutated into any of the symbols of the IUPAC code. When applying crossover, any pair of motifs has equal probability of being selected as parents. The genetic material of the parents is combined to produce an offspring. While the EA algorithm we

introduce in [21] allows for two-point or uniform crossover, the best results in [21] and the results in this work are obtained with the one-point crossover. While crossover is intended to mix the genetic code of each parent and confer it to fit children, the mutation operator is a fundamental evolutionary mechanism to provide diversity. Mutation is intended to prevent all offsprings/sought solutions of the fitness function to fall into a local optimum [7].

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 in an SVM are employed in the kernel function [33]. Given a k-mer w, the fitness value $f(w) = 100 * |c(w)_{\text{HS}}/\texttt{total}_{\text{HS}} - c(w)_{\text{non-HS}}/\texttt{total}_{\text{non-HS}}|$, where $c(w)$ counts the number of sequences containing w, and `total` normalizes by the number of known sequences in each class (HS or non-HS). According to this fitness function, a motif that is found in all HS sequences but no non-HS sequences, or alternatively in all non-HS sequences but no HS sequences, 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 [21,22] shows that the fitness value of the top motifs strongly correlates with the classification accuracy these motifs confer to an SVM.

In the results in section 3, an upper bound 5000 generations is used (convergence in the top fitness scores is generally obtained within the first 500 generations). The top 200 motifs of the final population are then employed to construct feature vectors from the input DNA sequences. These feature vectors need to be transformed by a kernel function. The GP algorithm proposed below searches the space of positive semi-definite kernel functions for the one that yields the highest SVM classification accuracy when applied to the feature vectors.

2.2 Genetic Programming Algorithm

The GP algorithm 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 GP algorithm.

Kernel Closure. Kernel functions must be continuous, symmetric, and preferably have a positive (semi-)definite Gram matrix [36]. Kernels that satisfy Mercer's theorem are positive semi-definite, meaning that their kernel matrices have non-negative eigenvalues. The use of a positive definite kernel insures that the optimization problem will be convex and the solution will be unique [36]. Work in [3] has demonstrated that kernels which are only conditionally positive definite can outperform most classical kernels in many applications. Moreover, new kernel functions can be constructed by combining well-known positive (semi)-definite kernel functions through specific mathematical manipulations that guarantee closure; that is, the new kernel functions inherit the properties of the kernel functions used to construct them [37].

Guaranteeing Closure. In this work, we employ an extensive list of kernel functions that have been proved to be positive (semi)-definite to construct new kernel functions that obey the closure property. The mathematical manipulations that allow doing so are listed in Table 2.

Table 2. Mathematical operations that guarantee closure. x, y refer to two feature vectors on which the kernel operates. k_1 and k_2 refer to two kernel functions used to construct a new kernel function k.

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:	$k(x,y) = e^{k_1(x,y)}$

Kernel Functions. Below is the extensive list of kernel functions that have been proved to be positive (semi)-definite and have been successfully employed in SVM classification. Our GP algorithm employs all of these kernel functions to evolve new ones.

(a) Linear Kernel. The linear kernel is the simplest kernel function. It is given by the inner dot product between two vectors and employs an optional coefficient c: $k(x,y) = x^T y + c$.

(b) Polynomial Kernel. The polynomial kernel is a non-stationary kernel well suited for problems where all the training data is normalized. The kernel contains parameters for slope α, coefficient constant c, and degree d: $k(x,y) = (\alpha x^T y + c)^d$.

(c) Gaussian Kernel. The Gaussian kernel is an example of the radial basis function kernel that is successfully employed as the default kernel in mainstream SVM implementations and applications. The adjustable parameter σ plays a major role in the performance of the kernel: $k(x,y) = e^{-\sigma \|x-y\|^2}$.

(d) Laplace Kernel. The Laplace kernel is another radial basis functional kernel, which is less sensitive to changes in the σ parameter: $k(x,y) = e^{-\frac{\|x-y\|}{\sigma}}$.

(e) Anova Kernel. The Anova kernel is another radial basis function kernel that performs well in multidimensional regression problems [16]: $k(x,y) = (\sum_{k=1}^{n} e^{-\sigma(x_k - y_k)^2})^d$.

(f) Sigmoid Kernel. The sigmoid kernel is popular due to its origin from neural network theory. Despite being only conditionally positive definite, it has been successful in various applications: $k(x,y) = \tanh(\alpha x^T y + c)$.

(g) Rational Quadratic Kernel. This kernel is often employed for faster computations to obtain similar results to the Gaussian kernel: $k(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2 + c}$.

(h) Inverse Multiquadratic Kernel. The Inverse Multiquadratic kernels has also been shown to be positive definite: $k(x,y) = \frac{1}{\sqrt{\|x-y\|^2+c^2}}$.

(i) Circular Kernel. The Circular kernel is inspired from statistics. It is an example of an isotropic stationary kernel and is positive definite in $\mathcal{R}^2$: $k(x,y) = \frac{2}{\pi}\arccos(-\frac{\|x-y\|}{\sigma}) - \frac{2}{\pi}\frac{\|x-y\|}{\sigma}\sqrt{1-\frac{\|x-y\|^2}{\sigma}}$.

(j) Spherical Kernel. The Spherical kernel is similar to the circular kernel, but it is positive definite in $\mathcal{R}^3$: $k(x,y) = 1 - \frac{3}{2}\frac{\|x-y\|}{\sigma} + \frac{1}{2}(\frac{\|x-y\|}{\sigma})^3$ if $\|x-y\| < \sigma$. Otherwise, $k(x,y) = 0$.

(k) Wave Kernel. The Wave kernel is also symmetric positive definite: $k(x,y) = \frac{\theta}{\|x-y\|}\sin(\frac{\|x-y\|}{\theta})$.

(l) Spline Kernel. The Spline kernel is given as a piecewise cubic polynomial and is positive semi-definite: $k(x,y) = \Pi_{i=1}^{d} 1 + x_i y_i + x_i y_i \min(x_i, y_i) - \frac{x_i+y_i}{2}\min(x_i,y_i)^2 + \frac{\min(x_i,y_i)^3}{3}$, where $x, y \in \mathcal{R}^d$.

(m) Bessel Kernel. The Bessel kernel is well known in the theory of function spaces of fractional smoothness. It is given by $k(x,y) = \frac{J_{v+1}(\sigma\|x-y\|)}{\|x-y\|^{-n(v+1)}}$, where J is a Bessel function of order 1.

(n) Cauchy Kernel. The Cauchy kernel is a long-tailed kernel that can be used to give long-range influence and sensitivity over a high-dimensional space of feature vectors: $k(x,y) = \frac{1}{1+\frac{\|x-y\|^2}{\sigma}}$.

(o) Chi-square Kernel. The Chi-square kernel comes from the well known chi-square distribution: $k(x,y) = 1 - \sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\frac{1}{2}(x_i+y_i)}$.

(p) Histogram Kernel. The Histogram Intersection kernel has been successfully used for image classification but is generally applicable to a variety of other applications: $k(x,y) = \sum_{i=1}^{n}\min(x_i, y_i)$.

(q) Generalized T-Student Kernel. The Generalized T-Student kernel has been proven to have a positive semi-definite matrix: $k(x,y) = \frac{1}{1+\|x-y\|^d}$.

Genetic Programming. Our GP algorithm simultaneously searches the space of kernel functions (and their parameters) and SVM cost parameters C. Each population consists of $2,000$ individuals. The first generation contains the extensive list of kernel functions detailed above, together with default values for their parameters and the SVM cost parameter C. A total of 30 generations are evolved to obtain the kernel function and cost parameter C that yield the highest classification accuracy (convergence in the fitness scores is observed after 20

generations). The fitness of each individual in a population is the classification accuracy obtained by applying the SVM on the feature vectors obtained with the above EA using the kernel function and parameter C contained in the individual.

Representation. Each individual is represented as a forest of two trees. One tree maintains the kernel function, and the other the cost parameter C. The tree that maintaints 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 non-terminal node in a kernel tree is either a mathematical operator (add, scalar, multiply, exponential) or a predefined kernel function (from the extensive list detailed above). Each terminal node in a kernel tree is either a kernel parameter (e.g., σ, d, θ) or one the input feature vectors x, y. See Fig. 1 for some examples of kernel functions.

Closure Constraint on Kernel Tree. Under the principle of closure, each node in the kernel tree may take any subtree as a child. In basic form, closure allows any non-terminal node to be a parent of any other node. We employ strongly typed GP (STGP) to evolve kernel functions. STGP places additional type constraints on the nodes, specifying the nodes that may link with others. STGP is typically used to allow child nodes to pass data to a parent that is guaranteed to be able to read the data [31]. This constraint allows the mutation and crossover operators to generate semantically-correct trees.

Ephemeral Constants. The cost parameter C and the various parameters of a kernel function (e.g., σ, d, θ) are terminal nodes internally implemented as ephemeral constants (ERC) [7] in our GP algorithm. All ERCs undergo mutation only. Parameters that take values in $\mathcal{R}$ are sampled uniformly at random from preset ranges each time they undergo mutation. For instance, the range of values for C is $[2^{-5}, 2^5]$. Parameters that take integer values (e.g., in the Anova and Bessel kernel functions) are internally implemented as ERC-int nodes. Table 3 gives a summary of all the non-terminal (mathematical operators and predefined kernel functions from the extensive list detailed above) and terminal nodes (ERC and ERC-int nodes representing kernel parameters and the SVM cost parameter C).

Mutation. The mutation operator is applied both to 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 ERC located in the subtree rooted at the selected node is then mutated according to a Gaussian probability distribution on over the preset range of the parameter maintained in the ERC node. When the selected tree is the kernel tree, the mutation operator changes the structure of the kernel function. Mutation is implemented through the growth method in [23]. A node picked at random from the kernel tree is replaced by a randomly-generated subtree, as illustrated in Fig. 1.

Table 3. All terminal and non-terminal nodes are shown

Name (Symbol)	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)	$k(x,y) = \frac{2}{\pi} \arccos(-\frac{\|x-y\|}{\sigma}) - \frac{2}{\pi}\frac{\|x-y\|}{\sigma}\sqrt{1 - \frac{\|x-y\|^2}{\sigma}}$	3,$\sigma \in \mathcal{R}$
Spherical (SPL)	$k(x,y) = 1 - \frac{3}{2}\frac{\|x-y\|}{\sigma} + \frac{1}{2}(\frac{\|x-y\|}{\sigma})^3$ if $\|x-y\| < \sigma$	3,$\sigma \in \mathcal{R}$
Wave (W)	$k(x,y) = \frac{\theta}{\|x-y\|}\sin(\frac{\|x-y\|}{\theta})$	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 (+)	$k(x,y) = k_1(x,y) + k_2(x,y)$	2Kernels
Multiply ($*$)	$k(x,y) = k_1(x,y) \cdot k_2(x,y)$	2Kernels
Scalar (Sc)	$k(x,y) = a \cdot k_1(x,y)$	1Kernel,$a \in \mathcal{R}$
Exponential (E)	$k(x,y) = e^{k_1(x,y)}$	1Kernel
$x_i, \ldots, x_n$		0,input
$y_i, \ldots, y_n$		0,input
ERC-int		Integer
ERC		Real

Crossover. the crossover operator applies only to the kernel tree. Parents are selected through the 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. Fig. 1 provides an illustration of crossover.

Bloat Control. One issue in GP is the unconstrained growth of individuals with no performance improvement. This growth, called bloat, may be limited by special genetic operators that restrict the maximum tree depth of an individual. An alternative technique, parsimony pressure, penalizes the size of an individual by making it less fit. We employ a simple technique, lexicographic tournament selection, to control bloat. Lexicographic tournament selection is similar to

tournament selection. The only difference is that, if multiple individuals have the same fitness, the individual with the shortest tree depth is chosen [30].

Putting it All Together. The optimal features are evolved separately through the EA we propose in [21] and employ here to convert input DNA sequences into feature vectors. The fitness of each individual in the 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. The data set with the input DNA sequences and methodology employed to measure classification accuracy are detailed below.

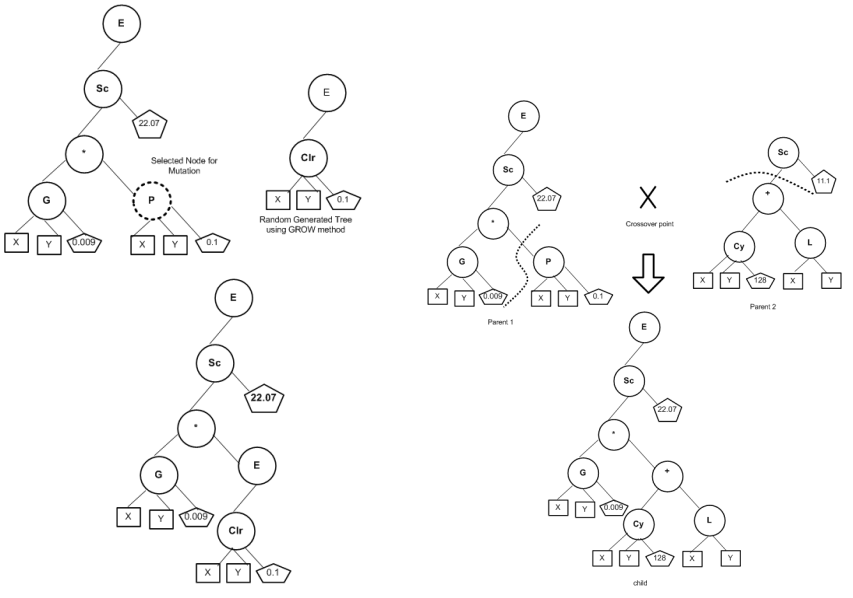

Fig. 1. Mutation (left) and crossover (right) are illustrated on sample kernel trees

Data Set. 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. The HS sequences were identified through a novel experimental methodology that employs cloning and in-vivo activity of K562 erythroid cells [35]. The non-HS sequences were not hypersensitive when tested in the same cell type. HS and non-HS sequences have similar average lengths of 242 nucleotides.

Test Methodology. The performance of the SVM is tested via 10-fold cross-validation on the total set of 1017 sequences (280 HS and 737 non-HS). 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 ROC curve is reported as an average over the 10-fold validations.

Implementation Details. 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 [27]. We use the open-source Biojava project [17] to integrate bioinformatics utilities for genomic sequences and the libSVM package [10] for the SVM implementation.

3 Results

The EA is run 30 different times to obtain 30 different sets of top motifs. 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 shown to outperform other predefined kernels in our recent work [21]. 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.

Table 4 shows how the classification accuracy changes when the kernel parameters are tuned with the grid search technique in [10]. Only three sets of feature vectors are tested, the worst, average, and best features, respectively associated with the minimum, average, and maximum accuracies obtained with the default RBF kernel. Reported values are averaged over the 10-fold cross-validation.

Table 4. Accuracies obtained with the tuned RBF kernel when employing worst, average, and best features

	Worst Features	Average Features	Best Features
Tuned Kernel	79.30	83.78	85.39

Table 5 shows the classification accuracies obtained when the tuned RBF kernel is replaced with the top kernel function and parameter values obtained by our GP algorithm. The values reported in Table 5 are averaged over the top kernel and parameters obtained from 30 different runs of our GP algorithm. Standard deviations vary from 0.2 to 0.4.

Table 5. Accuracies obtained when replacing the tuned RBF kernel with the evolved kernel and parameters reported by the GP algorithm

	Worst Features	Average Features	Best Features
Evolved Kernel	82.17	85.97	87.21

Finally, Table 6 shows some of the top individuals (kernel and cost parameters C) obtained. The fitness of each individual (SVM classification accuracy on best features vectors) is shown in the last column.

Table 6. Fitness values are shown for some of the top individuals (kernel function and cost parameter) obtained with our GP algorithm

Evolved Kernel	Cost Parameter	Fitness
$(*(CY(x)(y)(6.1394E-4))(S(x)(y)(7.4104004E-4)))$	17.85	87.35
$(CLR(x)(y)(10.746466))$	1.18	86.85
$(CY(x)(y)(2.3240509E-4))$	1.68	86.81
$(Sc(SPL(x)(y)(56.92857))(48.207794))$	1.00	86.79
$(*(CY(x)(y)(30.50425))(SPL(x)(y)(92.285255)))$	1.24	86.77

4 Conclusions

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 decisions in SVM, which often require previous expert knowledge or significant experimentation.

Some of the top kernel functions obtained by our GP algorithm are a combination of operators, such as exponential, multiplication,or scalar over predefined kernels, such as Gaussian, Sigmoid, and Linear. The presence of kernel functions, such as Cauchy, Circular, Spiral, and Student-T among top-scoring kernels further justify our employment of an extensive list of kernel functions in our GP algorithm. We emphasize that our GP algorithm maintains closure and guarantees the optimality of obtained kernels.

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.

References

1. Blanchette, M., Bataille, A.R., Chen, X., Poitras, C., Laganiere, J., Lefebvre, C., Deblois, G., Giguere, V., Ferretti, V., Bergeron, D., Coulombe, B., Robert, F.: Genome-wide computational prediction of transcriptional regulatory modules reveals new insights into human gene expression. Genome Res. 16(5), 656–668 (2006)

2. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Haussler, D. (ed.) 5th Annual ACM Workshop on COLT, pp. 144–152. ACM Press (1992)
3. Boughorbel, S., Tarel, J.-P., Boujemaa, N.: Conditionally positive definite kernels for svm based image recognition. In: Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2005), Amsterdam, The Netherlands (2005), http://perso.lcpc.fr/tarel.jean-philippe/publis/icme05.html
4. Burgess-Beusse, B., Farrell, C., Gaszner, M., Litt, M., Mutskov, V., Recillas-Targa, F., 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)
5. I. Committee. Nomenclature committee of the international union of biochemistry (nc-iub). nomenclature for incompletely specified bases in nucleic acid sequences. recommendations 1984. Biochemistry 229(2), 75–88 (1985)
6. Consortium IHGS. Finishing the euchromatic sequence of the human genome. Nature 431(7011), 931–945 (2004)
7. De Jong, K.A.: Evolutionary computation: a unified approach. MIT Press, Cambridge (2001)
8. de Souza, B.F., de Carvalho, A.C., Calvo, R., Ishii, R.P.: Multiclass svm model selection using particle swarm optimization. In: Sixth International Conference on Hybrid Intelligent Systems (2006)
9. Dorschner, M.O., Hawrylycz, M., Humbert, R., Wallace, J.C., Shafer, A., Kawamoto, J., Mack, J., Hall, R., Goldy, J., Sabo, P.J., Kohli, A., Li, Q., McArthur, M., Stamatoyannopoulos, J.A.: High-throughput localization of functional elements by quantitative chromatin profiling. Nat. Methods 1(3), 219–225 (2004)
10. 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)
11. Friedrichs, F., Igel, C.: Evolutionary tuning of multiple svm parameters. In: 12th European Symposium on Artificial Neural Networks (ESANN 2004), pp. 519–524 (2004)
12. Gagné, C., Schoenauer, M., Sebag, M., Tomassini, M.: Genetic Programming for Kernel-Based Learning with Co-evolving Subsets Selection. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 1008–1017. Springer, Heidelberg (2006)
13. Gross, D.S., Garrard, W.T.: Nuclear hypersensitive sites in chromatin. Annu. Rev. Biochem. 57, 159–197 (1988)
14. Habib, T., Zhang, C., Yang, J.Y., Yang, M.Q., Deng, Y.: Supervised learning method for the prediction of subcellular localization of proteins using amino acid and amino acid pair composition. BMC Genom. 9(suppl. 1), S1–S16 (2008)
15. Higgs, D.R., Vernimmen, D., Hughes, J., Gibbons, R.: Using genomics to study how chromatin influences gene expression. Annu. Rev. Genom. Human Genet. 8, 299–325 (2007)
16. Hofmann, T., Schölkopf, B., Smola, A.: Kernel methods in machine learning. The Annals of Statistics 36(3), 1171–1220 (2008)
17. Holland, R.C., Down, T.A., Pocock, M., Prlic, A., Huen, D., James, K., Foisy, S., Draeger, A., Yates, A., Heuer, M., Schreiber, M.J.: BioJava: an open-source framework for bioinformatics. Bioinformatics 24(18), 2096–2097 (2008)
18. Huang, C.-L., Wang, C.-J.: A ga-based feature selection and parameter optimization for support vector machines. Expert Systems with Applications, 231–240 (2006)

19. Islamaj-Dogan, R., Getoor, L., Wilbur, W.J.: A feature generation algorithm with applications to biological sequence classification. In: Liu, H., Motoda, H. (eds.) Computational Methods of Feature Selection. Springer, Berlin (2007)
20. Islamaj-Dogan, R., Getoor, L., Wilbur, W.J., Mount, S.M.: Features generated for computational splice-site prediction correspond to functional elements. BMC Bioinformatics 8, 410–416 (2007)
21. Kamath, U., De Jong, K.A., Shehu, A.: Selecting predictive features for recognition of hypersensitive sites of regulatory genomic sequences with an evolutionary algorithm. In: GECCO: Gen. Evol. Comp. Conf., pp. 179–186. ACM, New York (2010)
22. Kamath, U., Shehu, A., De Jong, K.A.: Using evolutionary computation to improve svm classification. In: WCCI: IEEE World Conf. Comp. Intel. IEEE Press (2010) (in press)
23. Koza, J.: On the Programming of Computers by Means of Natural Selection. MIT Press, Boston (1992)
24. Leslie, C., Kuang, R., Bennett, K.: Fast string kernels using inexact matching for protein sequences. Journal of Machine Learning Research 5, 1435–1455 (2004)
25. Leslie CS, N.W., Eskin E.: The spectrum kernel: a string kernel for svm protein classification. In: Pacific Symposium on Biocomputing, Baoding, China, vol. 7, pp. 564–575 (2002)
26. Lowrey, C.H., Bodine, D.M., Nienhuis, A.W.: Mechanism of DNase I hypersensitive site formation within the human globin locus control region. Proc. Natl. Acad. Sci. USA 89(3), 1143–1147 (1992)
27. Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Popovici, E., Sullivan, K., Harrison, J., Bassett, J., Hubley, R., Chircop, A., Compton, J., Haddon, W., Donnelly, S., Jamil, B., OBeirne, J.: ECJ:Ajava-based evolutionary computation research (2010)
28. Staelin, C.: Parameter Selection for Support Vector Machines, Internal publication of HP Laboratories, Israel (approved for external publication) Technion City, Haifa, 32000. Israel Copyright Hewlett-Packard Company (2002), `http://www.hpl.hp.com/techreports/2002/HPL-2002-354R1.pdf`
29. Maston, G.A., Evans, S.K., Green, M.R.: Transriptional regulatory elements in the human genome. Annu. Rev. Genom. Human Genet. 7, 29–59 (2006)
30. Mierswa, I.: Evolutionary learning with kernels: A generic solution for large margin problems. In: GECCO: Gen. Evol. Comp. Conf., pp. 1553–1560 (2006)
31. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1993)
32. Noble, W.S.: Support vector machine applications in computational biology. In: Schölkopf, B., Tsuda, K., Vert, J.-P. (eds.) Kernel Methods in Computational Biology. MIT Press, Cambridge (2004)
33. Noble, W.S., Kuehn, S., Thurman, R., Yu, M., Stamatoyannopoulos, J.A.: Predicting the in vivo signature of human gene regulatory sequences. Bioinformatics 21(suppl. 1), i338–i343 (2005)
34. Phienthrakul, T., Kijsirikul, B.: Evolutionary strategies for multi-scale radial basis function kernels in support vector machines. In: Genetic and Evolutionary Computation Conference, Washington D.C.,USA, pp. 905–911 (2005)
35. Sabo, P.J., Humbert, R., Hawrylycz, M., Wallace, J.C., Dorschner, M.O., McArthur, M., Stamatoyannopoulos, J.A.: Genome-wide identification of DNase I hypersensitive sites using active chromatin sequence libraries. Proc. Natl. Acad. Sci. USA 101(13), 4537–4542 (2004)

36. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Boston (2002)
37. Shawe-Taylor, J., Cristianini, N.: Kernel methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
38. Sullivan, K., Luke, S.: Evolving kernels for support vector machine classification. In: Genetic and Evolutionary Computation Conference (2007)
39. Vapnik, V.N.: Statistical learning theory. Wiley & Sons, New York (1998)
40. Vertanen, K.: Genetic adventures in parallel: Towards a good island model under PVM (1998)
41. Wu, C.: The $5'$ ends of drosophila heat shock genes in chromatin are hypersensitive to DNase I. Nature 286(5776), 854–860 (1980)
42. Zhang, X.H., Heller, K.A., Hefter, I., Leslie, C.S., Chasin, L.A.: Sequence information for the splicing of human pre-mrna identified by support vector machine classification. Genome Res. 13(12), 2637–2650 (2003)
43. Zien, A., Raetsch, G., Mika, S., Schölkopf, B., Lengauer, T., Mueller, K.R.: Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics 16(9), 799–807 (2000)