Terminator Detection by Support Vector Machine Utilizing a Stochastic Context-Free Grammar

Patricia Francis-Lyon, Nello Cristianini: University of California at Davis
Stephen Holbrook: Lawrence Berkeley National Laboratory

Abstract

A 2-stage detector was designed to find rho-independent transcription terminators in the Escherichia coli genome. The detector includes a Stochastic Context Free Grammar (SCFG) component and a Support Vector Machine (SVM) component. To find terminators, the SCFG searches the intergenic regions of nucleotide sequence for local matches to a terminator grammar that was designed and trained utilizing examples of known terminators. The grammar selects sequences that are the best candidates for terminators and assigns them a prefix, stem-loop, suffix structure using the Cocke-Younger-Kasaami (CYK) algorithm, modified to incorporate energy affects of base pairing. The parameters from this inferred structure are passed to the SVM classifier, which distinguishes terminators from non-terminators that score high according to the terminator grammar. The SVM was trained with negative examples drawn from intergenic sequences that include both featureless and RNA gene regions (which were assigned prefix, stem-loop, suffix structure by the SCFG), so that it successfully distinguishes terminators from either of these. The classifier was found to be 96.4% successful during testing.

Introduction

Two types of transcription terminators, named for their operating mechanisms, have been found to exist in bacteria: rho-dependent and rho-independent terminators. Detection of terminators has been challenging due to the lack of clear signals in their genetic sequence, such as is provided to protein gene detection by start and stop codons. However, there are structural features present in the class of rho-independent terminators that may be exploited to aid in their detection.

For a rho-independent terminator, the ability to function effectively is largely due to formation of a stem-loop. This secondary structure, rather than sequence, is the phenotype selected for in the evolutionary process. The same structures may result from different sequences of nucleotides adenine, cytosine guanine and uracil (a,c,g and u). Therefore sequences may be evolutionarily related while not conserved, as long as their structures are conserved by compensatory mutations. (For example, a stem cg pair can be replaced by gc, au, ua gu or ug pair.) Unsurprisingly, it has been found that rho-independent terminators do not share general consensus sequence [1]. Our approach to terminator detection is to infer structural information from sequence alone, then use both sequence and inferred structural parameters to classify the sequence as terminator or non-terminator.

Background

Terminator Detection

Transcription is the process by which a copy of the coding (nontemplate) strand of a gene is produced, except that thymine (t) in DNA is replaced by uracil (u) in RNA, resulting in an RNA transcript. The final phase of transcription is termination, which can be signaled in rho-independent terminators by the formation of a stem-loop within the RNA polymerase (RNAP), inducing the pausing of the transcription elongation complex (TEC) just as the RNAP encounters weak au bonds at the terminator tail, causing the dissociation of the TEC from the RNAP and the release the protein or RNA gene.

A model attributed to Carafa et al [2] describes DNA sequence for rho-independent terminators. An RNA hairpin (stem-loop) is followed by a 15 nucleotide (nt) long region rich in thymidine (the nucleoside of thymine) which may be separated by a spacer region of up to 2 nts. An adenoside-rich region was described upstream of the hairpin (but not used in their scoring system). Fig. 0 depicts the canonical terminator, based upon the Carafa model, that was used in this project. Carafa et al developed a 2-stage process to detect and classify candidate terminators which takes into account structural information such as free energy of the RNA hairpin, along with stem and loop length. Sequence information such as the number and positions of thymidine residues, and the fraction of cg pairs in the stem is also used. This algorithm

successfully distinguishes between terminators and both random sequence and protein coding sequence.

Other researchers have built upon the Carafa model to create terminator detectors. The 2-stage process of Ermolaeva et al [3] utilized location and orientation information, their own representation of the stability of stem-loop structure, and the Carafa tail-scoring function. Lesnik et al [4] devised an algorithm utilizing sequence parameters and allowing the user to define constraints upon structure. Their thermodynamic scoring system accounts for the preference of stem-loop structure over bonding to DNA at the point of transcription termination. More recently, de Hoon et al [5] used a logistic regression model to arrive at a decision rule for predicting rho-independent terminators in B. subtilis and related species. While these methods detect more known terminators than Carafa et al, they report a tradeoff between finding more known terminators (true positives) and getting more false positives. Also, this sensitivity/specificity tradeoff is hard to quantify since many terminators have yet to be experimentally determined so numbers of true and false positives in these studies are estimates. Some studies count the terminators found by their algorithm as true positives along with experimentally determined terminators as long as these putative terminators satisfy location and scoring standards. Ermolaeva et al, who create a polynomial approximation to estimate frequency of false positives, report finding 567 terminators in E. coli with specificity of 98%. This specificity indicates that the authors regard 555 of the terminators found by their algorithm as true terminators, which is far higher than the number of experimentally determined terminators. At this specificity they reportedly find 89% of all true terminators (sensitivity).

The success of Hidden Markov models in statistical modeling, database searching and multiple alignment of both promoters and protein genes has prompted researchers to look to grammars to incorporate long range interactions into feature detection. Hidden Markov models are equivalent to stochastic regular grammars, where sequence is generated from left to right [6]. Features that have stem-loop structures caused by base pairs that are nested can be generated by a context-free grammar, emitting sequence from outside to inside rather than from left to right. Stochastic context-free grammars (SCFGs) capture both sequence and structural information and have been used successfully to model RNA genes by Eddy and Durbin [7] and tRNA genes by Sakakibara et al [8]. SCFGs were used to model terminators by Bockhorst and Craven [9], as a test case to show that a deficient SCFG could be refined in an iterative process. Their paper states that preliminary results indicate that the refinement method produced an SCFG that improved the accuracy of the model, but the success rates themselves were not reported.

For this project, a Stochastic Context Free Grammar (SCFG) was developed to utilize both sequence and structural information to detect terminators from genomic sequence. To refine the detection process, a support vector machine (SVM) was coupled with the SCFG. The SCFG selects likely candidates for terminators from sequence, and designates subsequences of each as prefix, stem, loop and suffix. This information is passed to the SVM, which was trained to distinguish between terminators and those non-terminators that were assigned high scores by the SCFG. The terminator grammar of Bockhorst and Craven was not used, because the grammar was reported to be deficient and also involves a trifurcation that is computationally intensive. Rather, a grammar was developed to select sequence that can take on the structure of the Carafa canonical rho-indedpendent terminator.

Stochastic Context-Free Grammars

A Grammar is a set of rules, called productions, together with a set of abstract symbols called non-terminals and a set of emitted symbols called terminals. Together these 3 sets characterize the set of legal strings, the language of the grammar, which are those strings that can be derived by iterative application of the productions. Grammars having a set of terminals consisting of {a,c,g,t} have been used for modeling strings of nucleotides, such as genes.

The structure of a stochastic context-free grammar is that of its underlying context-free grammar G. G can be formally defined as G={**N**,**T**,**P**,*S*} where **N** is a finite set of nonterminal symbols ("states"), **T** is a finite set of terminal symbols ({a,c,g,t} for nucleotides), **P** is a finite set of productions of the form $\mathbf{A} \rightarrow \mathbf{\Gamma}$, where $\mathbf{A} \in \mathbf{N}$, $\mathbf{\Gamma} \in (\mathbf{N} \cup \mathbf{T})^*$, and S is the start nonterminal ($S \in \mathbf{N}$). This means that the right hand side of a production may consist of any combination of terminals and nonterminals, while the left hand side must be a single nonterminal. A particular iterative application of productions that result in a string x is referred to as a derivation or may be viewed as a parse tree, Π, for x.

An SCFG assigns a probability to each production rule in P such that all the productions from any given nonterminal sum to 1. The set of these probabilities is referred to as the parameters of the model, θ. A sequence x may have a higher probability (score) with one model than with another. The probability $P(x,\Pi \mid G,\theta)$ is the probability that a particular derivation (parse tree) Π generates string x given structure G of the underlying context-free grammar and the probabilities θ associated with the productions. This is simply the product of all the production rules used in the parse tree Π for sequence x. An SCFG describes a joint probability distribution $P(\mathbf{x},\mathbf{\Pi} \mid G,\theta)$ over all sequences **x** and all possible parse trees **Π**.

SCFGs are generalizations of hidden Markov models (HMMs), which are equivalent to stochastic regular grammars. In addition to primary sequence, modeled in left to right string generation by HMMS, SCFGs model secondary structure in outside to inside generation of strings. The HMM algorithms for solving problems of detection, alignment and parameter

estimation have analogs for families modeled by SCFGs. These dynamic programming algorithms start with subsequences of length zero and consider larger and larger sequences by incrementally extending them. In the case of HMMs, subsequences are extended leftwards by 1 nt at a time, whereas for SCFG models, subsequences are extended outwards by 2 nt at a time, capturing pair interactions.

For this project, genomic sequence needs to be parameterized into likely terminator structure so the SVM could be trained to detect which are terminators. To accomplish this the Cocke-Younger-Kasaami (CYK) algorithm is used for alignment to the structure and model. Rather than taking the sum of probabilities of parse trees as is done in scoring, these alignment algorithms find the argmax of the probabilities for parse trees. The end result is $\log P(x,\Pi^* \mid G,\theta)$ where Π^* is the most likely parse for x given the grammar structure and model. A traceback can be coded to reveal Π^*.

Support Vector Machines

Support Vector Machines (SVMs) are a relatively recent addition to the field of machine learning. They were introduced by Vapnik and began to be widely used in classification in the 1990's. SVMs are trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory to search a hypothesis space of linear functions operating on data that has been pushed into a high dimensional feature space [10]. Basically, an SVM is a hyperplane classifier which finds the optimal hyperplane to separate data into classes. When dividing two classes, the optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes, and intersecting it halfway between the two classes at a perpendicular distance d from either class, creating a margin of 2d between the classes. The support vectors are those elements of the training set that lie on the margins of either class (at a distance d from the separating hyperplane). It is these training examples that are relevant to the algorithm. It is these training examples, rather than the centers of clusters, that are critical for finding the margins between the classes. Complexity of the algorithm may be reduced by removing the other training examples from the kernel expansion. It can be shown by geometry that the margin we want to maximize equals $2/||\mathbf{w}||^2$, so the unique optimal hyperplane is found by solving the optimization problem:

$$\text{Minimize} \quad T(\mathbf{w}) = \tfrac{1}{2}\,||\mathbf{w}||^2 \qquad (1)$$

$$\text{Subject to} \quad y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) >= 1, \quad i = 1,2,\ldots,m$$

where $||\mathbf{w}||^2$ is the norm of the separating hyperplane and $\mathbf{x}_i$ is the n dimensional vector representing the ith data point of m data points. The minimization of this optimization problem is solved using Lagrange multipliers and minimizing the Lagrangian.

SVMs have the ability to find a separating hyperplane even if one does not exist in the space of the input vector, as long as the training data may be mapped into a higher dimensional feature space in which such a separating hyperplane exists. The kernel function is used to compute the separating hyperplane without actually having to carry out the mapping into higher dimensional space.

To allow for noise in the data that would preclude perfect classification, a slack variable can be introduced in order to relax the constraints to :

$$\text{Subject to} \quad y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) >= 1 - e_i, \quad i = 1,2,\ldots,m \qquad (2)$$

$$\text{Where slack variables} \quad e_i >= 0, \quad i = 1,2,\ldots,m$$

The amount of slack is specified by the user of an SVM in the variable C, which gives the upper bound on the Lagrange multipliers in the optimization problem. A lower value of C limits the influence of outliers on the solution.

Choice of kernel determines both the class of functions from which the solution is taken and the type of regularization that is used in minimizing the regularized risk, which limits the complexity of a function class, helping to avoid overfitting. The common kernels used are Gaussian RBF, polynomial, sigmoidal, and inverse quadratic [12].

Methods

Terminator Grammar

Because protein genes are generally easy to detect due to the presence of start and stop codons, this project incorporated the often-used strategy of searching only those regions between protein genes to detecting features such as terminators. A crucial requirement is that the detector be able to distinguish terminators from RNA genes, which also have stem-loops.

The terminator structure modeled by the grammar is prefix, followed by stem-loop, followed by suffix (Fig. 2). The grammar is simple, using software rules to enforce requirements such as minimum stem length, rather than complicating the grammar. Also rather than using a computationally intensive trifurcation from the start state into 3 states of prefix, stem-loop and suffix as Bockhorst and Craven do, this terminator grammar uses a simpler emissions-based approach that requires only a bifurcation. Productions for the terminator grammar are:

$S \rightarrow L\ R$
$R \rightarrow Rb \mid P$
$P \rightarrow aPa' \mid L$
$L \rightarrow bL \mid end$

where S is the start state, R is the state for rightwise emission, P is the state for pairwise emission and L is the state for leftwise emission.

Fig. 0. Canonical Terminator

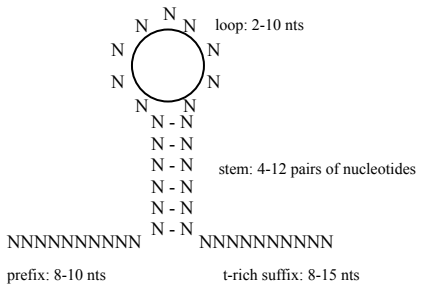
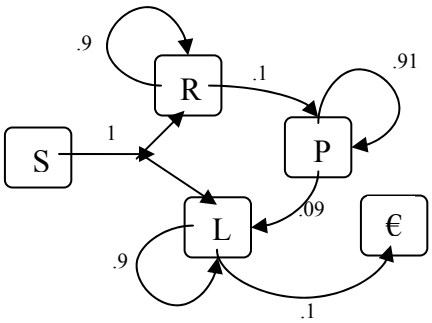


Fig 1. State Diagram for Terminator SCFG



A state diagram is represented in Fig. 1, including transition probabilities. Probability matrices for single emission and pairwise emission complete the model. A terminator can be modeled using only leftwise (or rightwise) single-emission. However, the outside to inside generation of stem-loop requires an end state from the loop, as well as from either the prefix or suffix. Also, only one transition into the stem-loop is allowed in a terminator, so a single-emission state separate from the loop state is required. This necessitates 2 single-emission states, However, both of them could just as easily emit either rightwise or leftwise, as with the alternative grammar of Fig. 2.

For the purposes of illustrating the grammar, an overly simple "toy" terminator (with only 2 nts for prefix, stem length, loop and suffix) is used in Fig.3 (derivation) and Fig. 4 (parse tree). Fig. 5 illustrates the CYK algorithm that was written to scan large sections of genome for local matches to the grammar. Although the toy terminator of Fig. 2-5 is only 10 nts (too short to be an actual terminator) and the match is global to simplify the illustration, the intergenic stretches of genome searched may be longer than 5000 nts. For this reason the matrix structure used by the CYK algorithm of this project is a variant [6] of the familiar LxL structure, which would require $O(ML^2)$ space and $O(ML^3)$ time. Instead, the entire string of length L is searched for local matches of at most length D = min{L, max terminator length}. D is therefore capped at 70, which shortens the space requirements of CYK algorithm to O(MLD) and time complexity to $O(MLD^2)$.

The (LxD) matrix structure is depicted in Fig. 5, one of which is required for each of the M states to make up the full three dimensional data structure. Each of the MLD cells takes time O(1) (proportional to the number of transitions from the state) to fill, except for state S, which has a bifurcation and so takes O(D) to fill, resulting in total running time of $O(MLD^2)$. The score resulting from the CYK algorithm coded for this project gives a different result that than the familiar log $P(x,\Pi^* \mid G,\theta)$, because a value was added for base pairing (as in the Zuker algorithm [7]), to account for energy of folding affects.

The parameters of the grammar model were trained using a randomly selected subset of the known terminators. Initial priors for emissions and transitions were established by statistical analysis of the training subset as folded by a locally developed rule-based algorithm. This initial model was then used by the grammar to select a parse (prefix, stem, loop, suffix structure) for each terminator. Counts were taken of frequencies of emissions and transitions, the model parameters were updated based upon these counts. The process was iterated until structure was stable.

Software and Datasets

The Support Vector Machine toolbox for Matlab, Version 2.51, was used to construct and train the SVM that classified the data [12]. Locally developed software produced negative examples, implemented a modified CYK algorithm, trained the terminator SCFG and calculated parameters. Datasources were the complete E coli genome sequence and annotations compiled at the University of Wisconsin-Madison [13], which are available on the web. The sequence used was the M54 version of the K-12 MG1655 strain of E. coli. The sequences used for positive examples were all 109 known terminators found within -10 to 60 nts from the end of a gene as documented by Lesnik et al. Negative examples were drawn from both strands of the genome. This was accomplished by starting with the entire strand and removing known genes of either strand as well as a trailing region of 300 nts in order to minimize the possibility that unknown terminator sequence would comprise part of the set of negative training examples. The sequences of the 168 known RNA genes were added to the intergenic sequences to form the set of sequences from which negative examples were drawn. Some of the sequences were very long (up to 5000 nts) and contained many local matches to the terminator grammar.

Fig. 2. Alternative Terminator Grammar

$S \rightarrow L\ X$
$L \rightarrow bL \mid P$
$P \rightarrow aPa` \mid X$
$X \rightarrow bX \mid end$

Fig. 2. Structure of Simplified Terminator

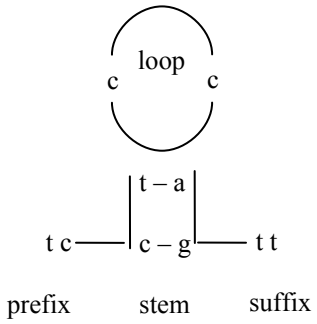


Fig. 4. Parse Tree for Derivation of "tcctccagtt"

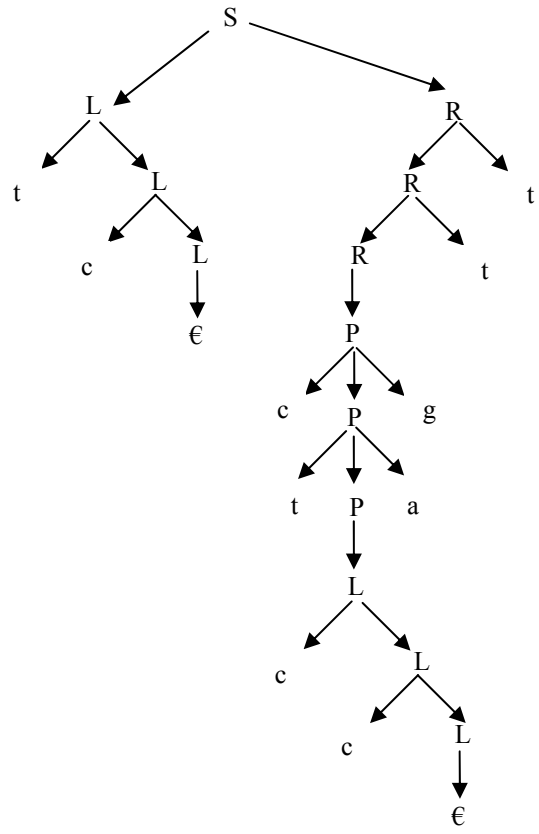


Fig. 3. Derivation of Simplified Terminator

S → L R
→ t L R
→ t cL R
→ t c€ R
→ t c R t
→ t c R t t
→ t c P t t
→ t c cPg t t
→ t c c tPa g t t
→ t c c t L a g t t
→ t cct cL a g t t
→ tcct c cL a g t t
→ tcct c c€ a g t t
= tcctccagtt

Fig. 5. State path taken by global CYK algorithm performed on string "tcctccagtt"

	d: 1	2	3	4	5	6	7	8	9	10
j: 1 t										
2 c	L	L **								
3 c										
4 t										
5 c										
6 c	L	P→ L								
7 a				P						
8 g						R → P				
9 t							R			
10 t								R **		Score=S

Collapsed view of the 4 arrays of the data structure which together form the 3 dimensional array structure for CYK algorithm
Coordinate system: end position j vs subsequence length d, d = j–i+1 for subsequence(i:j) being considered
Horizontal (green) run is leftwise emission (prefix of length 2 and loop of length 2): shows path through green array (state L)
Knight's path (red) run is for pairwise emission (stem of length 2): shows path through red array (state P)
Diagonal (blue) run is for rightwise emission (suffix of length 2): shows path through blue array (state R)
Global score considers entire string = subsequence(1:10) where j=10, d=10: shows path through yellow array (state S)

** S → LR at substring(1:10), adding scores from L of substring(1:2) and R of substring(3:10)

Experiment

For each sequence processed by the SCFG, the best local alignments (optimal parses) were determined by the CYK algorithm. This is, in effect, the selection of prefix, stem-loop and suffix structures for candidate terminators. A subset of 81 known terminator sequences was randomly selected for training the parameters of the grammar model, as described in the Methods section. After training, these sequences were processed by the SCFG for likely parses of prefix, stem-loop and suffix structures. Each of these sequences contained one likely parse which was parameterized, resulting in a total of 81 positive examples with which to train the SVM. The remaining 28 positive sequences that had not been used in training the SCGF were then processed into 28 positive examples with which to test the SVM. All 168 known RNA genes and the featureless intergenic sequences were processed by the SCFG into negative examples.

From the prefix, stem-loop and suffix structure inferred by the CYK algorithm for each example, 17 parameters were extracted. Three parameters represented fractions of a,c, and t in the prefix. Also extracted was the probability that the stem base pair closing the loop of the example would close the loop of an actual terminator (as calculated by their frequency counts in the 81 training sequences as folded by the SCFG). The frequencies of the 6 dinucleotide pairs in the stem that were found to aid detection (at cg ta tg gc gt) were extracted from the sequence being parameterized. The number of nucleotides between the stem and the first t following it was used, along with the tail scoring function of Carafa et al summed over all the nucleotides of the prefix:

$$T = \Sigma x_n \text{ where } \quad x_n = x_{n-1} * 0.9 \text{ if nth nucleotide is a t} \quad (3)$$
$$x_n = x_{n-1} * 0.6 \text{ if nth nucleotide is not a t}$$

Additionally there were 4 structural parameters: prefix length, stem length, loop length and suffix length. The SCFG score of the alignment was included, resulting in the 17 dimensional input into the SVM for each example being classified. All parameters were normalized.

Since there were many more negative than positive examples, negative intergenic and negative RNA gene examples were each randomly sampled to select the negative examples for the training and test sets, allowing 5 times as many negative as positive (terminator) examples, proportionately representing featureless intergenic and RNA gene examples. The training set was therefore comprised of the 81 terminators that had been used to train the SCFG, along with negative examples comprised of 376 featureless intergenic examples and 29 RNA genes. SVMs with radial basis, polynomial and linear kernels were constructed and tested. The most successful SVM utilized a linear kernel, with C (the upper bound on the Lagrange multipliers, as discussed above) set to 10, limiting the influence of outliers.

The test set was comprised of the 28 terminators that not been used to train the SCFG, along with 130 featureless intergenic examples and 10 negative RNA gene examples. An SVM was trained using the entire training set and tested on the test set. 96.4% were correctly identified, with detection of negative examples more successful than detection of positives. (table 1). Cross-validation was also used on the training set, resulting in 3 SVMs, which were then tested with the original test set. Results ranged from 97.0.5% to 97.6% correctly identified (table 2), with positive examples being detected almost as successfully as negative example (average 95.2%). Average prediction accuracy during cross-validation was 96.1% with average accuracy of RNA genes prediction of 97.0% (table 3).

Discussion

There are 2 major aspects to this detection method: 1) the selection of likely candidates from the genome and their parameterization by the SCFG based upon inferring structure as well as upon sequence composition and 2) the use of an SVM to detect terminators from these high scoring candidates utilizing the extracted features.

The success rates reported in tables 1-3 give a measure of how well the SVM accomplished the classification that it was trained to do. It is worthwhile to also evaluate each component of the 2-stage The SCFG retained local matches that scored above the cutoff, resulting in all known terminators being selected as likely terminators, along with 1063 more from among the negative examples. As described in the Background section, there is a sensitivity/specificity tradeoff: if the SCFG were to function alone as a detector, the cutoff score would be raised to have fewer false positives but reduced sensitivity of about 89% as with the tradeoff selected by Ermolaeva et al. Instead, cutoff was selected to allow 100% sensitivity for the SCFG alone, allowing the SVM to refine the prediction. After passing these candidates through SVM, the sensitivity of the 2-stage terminator detector was reduced to 92.9%, but the number of false positives went down to 4 (assuming none of the negative examples are as-yet-undiscovered terminators). A Receiver Operating Characteristics (ROC) curve is perhaps the best way to view the full situation of tradeoff between sensitivity and specificity. If the area under the ROC curve is 0.9 or above, the classifier is considered good. As shown in Fig. 6, the SCFG alone has good performance: the area under ROC = 0.97, but when coupled with the SVM, performance is better: 0.99.

Because of the structural similarity of RNA genes and terminators, the successes of both the SCFG and SVM components in distinguishing them was of interest. The SCFG was about as effective in filtering out non-terminator sequence from RNA genes as from featureless intergenic sequence. However, performance of the SVM component was better for terminator candidates drawn from intergenic than RNA gene sequence, indicating room for improvement in the SVM.

Future Work

The SCFG-SVM terminator algorithm would be more useful if a detector trained using examples from a well-annotated species could be used on an evolutionarily related but newly sequenced species with no known terminators. Future work would include testing this transferability by using the E. coli terminator detector on a different, but closely related species whose terminators are known.

The SCFG component of the detector might work better if the loop state was separate from the prefix state, allowing for different emissions parameters. Another way to improve the SCFG might be to change the grammar structure to allow for single emission in the middle of the terminator stem, which is a part of the Lesnik model left out of this grammar in order to simplify it.

Different approaches might be tried to address the far greater numbers of negative examples than positive examples provided to the SVM. The approach of Meraz et al (2004)* might improve SVM performance by using successive iterations of a support vector machine to select improved current negative sets by selecting for maximum dissimilarity to the positive set but also maximum distance to the negative set. The net affect is that the SVM decision boundary shrinks closer and closer to the positive set. Alternatively, it might improve results to use different C values for positive and negative examples such that the C value for positive examples is higher, weighting them more in the classification process. Also, the SVM might be better at detecting RNA genes if a greater proportion of the negative examples were drawn from RNA genes.

Conclusion

The application of machine learning methods in the detection of terminators poses a challenge due to the nature of their evolutionarily conserved phenotype, which is the secondary structure that the RNA takes on during transcription. The challenge is that given only sequence information, something must be inferred about the structure of the RNA. While a SCFG performed reasonably well by itself in inferring structure and detecting terminators, selection was much improved by raising the score cutoff to allow many terminator candidates, which were then classified by an SVM.

Table 1: Test Results of SVM Terminator Classifiers

	terminator correct	terminator wrong	neg ex correct	neg ex wrong	RNA genes correct	RNA genes wrong	total correct	total wrong
SVM	0.9286	0.0714	0.9714	0.0286	0.8000	0.2000	0.9643	0.0357

Table 2: Results of test set on Cross validation SVM Terminator Classifiers

	terminator correct	terminator wrong	neg ex correct	neg ex wrong	RNA genes correct	RNA genes wrong	total correct	total wrong
SVM 1	0.9286	0.0714	0.9786	0.0214	0.8	0.2	0.9702	0.0298
SVM 2	0.9643	0.0357	0.9786	0.0214	0.8	0.2	0.9762	0.0238
SVM 3	0.9643	0.0357	0.9786	0.0214	0.8	0.2	0.9762	0.0238
AVE	0.9524	0.0476	0.9786	0.0214	0.8	0.2	0.9742	0.0258

Table 3: Results of Cross validation for SVM Terminator Classifiers

	terminator correct	terminator wrong	neg ex correct	neg ex wrong	RNA genes correct	RNA genes wrong	total correct	total wrong
SVM 1	0.9259	0.0741	0.9850	0.0150	1.0000	0.0000	0.9750	0.0250
SVM 2	0.8889	0.1111	0.9774	0.0226	1.0000	0.0000	0.9625	0.0375
SVM 3	0.8889	0.1111	0.9568	0.0432	0.9091	0.0909	0.9458	0.0542
AVE	0.9012	0.0988	0.9731	0.0269	0.9697	0.0303	0.9611	0.0389

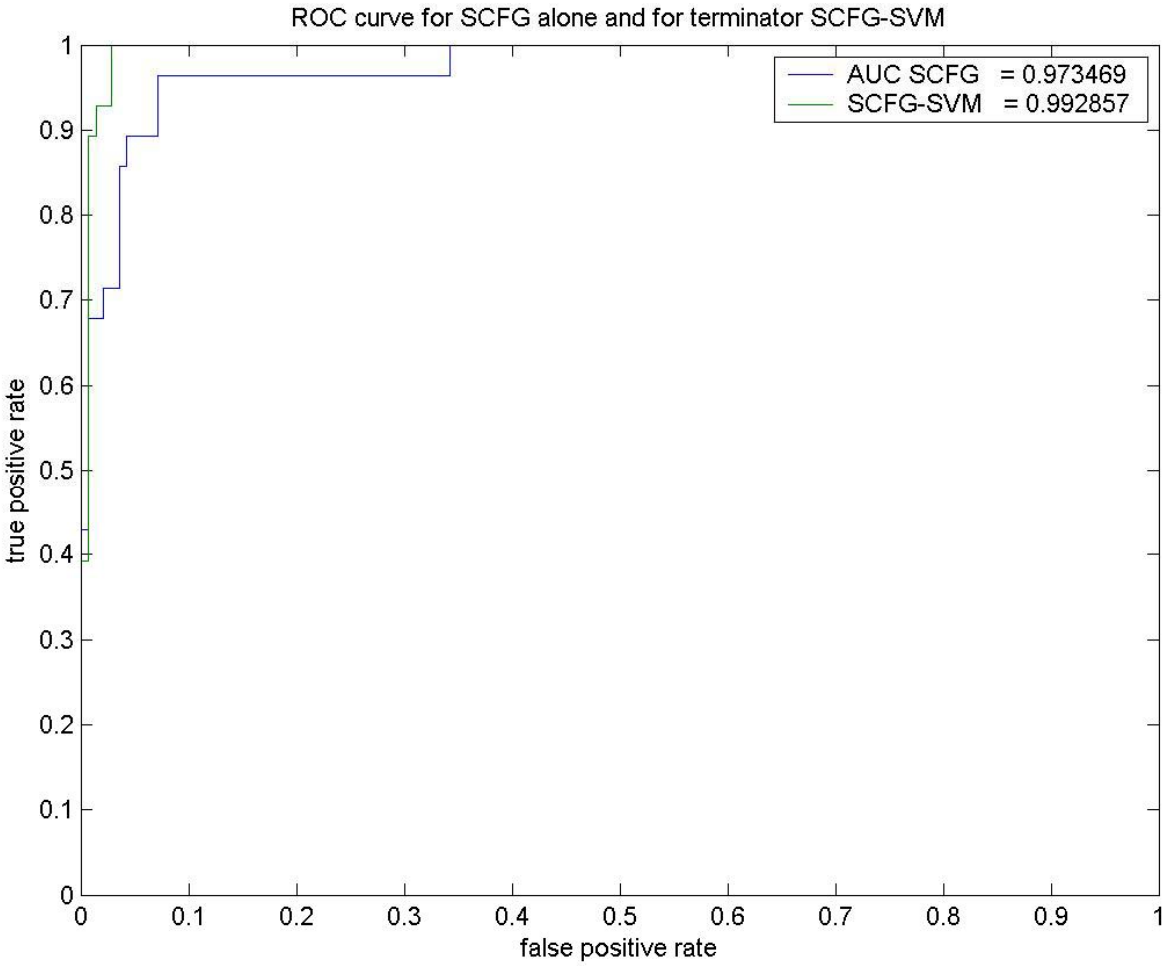


Fig. 6. ROC curves show good detection with SCFG alone, but better with 2-stage detector

REFERENCES

[1] Platt, T. (1966) Transcription termination and the regulation of gene expression. *Annu. Rev. Biochem*. 55, 339-372.

[2] Carafa Y., Brody E. and Thermes, C. (1990) Prediction of Rho-independent *Escherichia coli* Transcription Terminators: A Statistical Analysis of their Stem-Loop Structures. *J. Mol. Bio.* 216, 835-858.

[3] Ermolaeva, M., Khalak H., White O., Smith H. and Salzberg S. (2000) Prediction of Transcription Terminators in Bacterial Genomes. *J. Mol. Bio.* 301, 27-33.

[4] Lesnik E., Sampath R., Levene H., Henderson T., McNeil J. and Ecker, D. (2001) Prediction of rho-independent transcriptional terminators in *Escherichia coli*. *Nucleic Acids Research* 29:3583-3594.

[5] de Hoon M.,Makita Y., Nakai K. and Miyano S. (2005) Prediction of Transcriptional Terminators in *Bacillus subtilis* and Related Species. PLoS Comput Biol 1(3): e25.

[6] Durbin,R.., Eddy S., Krogh A. and Mitchison G. (1998) Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press.

[7] Eddy S., and Durbin R. (1994) RNA sequence analysis using covariance models. *Nucleic Acids Research* 22:2079-2088.

[8] Sakakibara Y., Brown M., Hughey R., Mian I.S., Sjolander K.., Underwood R.C. and Haussler D. (1994) Stochasitc context-free grammars for tRNA modeling. *Nucleic Acids Research* 22:5112-5120.

[9] Bockhorst J. and Craven M. (2001) Refining the Structure of a Stochastic Context-Free Grammar. *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (IJCAI 2001).

[10] Cristianini N., and Shawe-Taylor J. (2000) An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press.

[11] Müller, K., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (March, 2001) An Introduction to Kernel–Based Learning Algorithms. *IEEE Transactions of Neural Networks*, vol 12, no. 2.

[12] Schwaighofer, Anton (January 2002) Support Vector Machine toolbox, Version 2.51. GNU public license.

[13] Blattner,F.R., Plunkett,G., Bloch,C.A., Perna,N.T., Burland,V., Riley,M., Collado-Vides,J., Glasner,J.D., Rode,C.K. and Mayhew,G.F. (1997) The complete genome sequence of *Escherichia coli* K-12. *Science*, **277**, 1453–1474. Datasources available on the web at: http://www.genome.wisc.edu/sequencing/k12.htm