USING SIGNAL PROCESSING, EVOLUTIONARY COMPUTATION, AND MACHINE LEARNING TO IDENTIFY TRANSPOSABLE ELEMENTS IN GENOMES

WENDY COLE ASHLOCK

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO
AUGUST 2013

USING SIGNAL PROCESSING, EVOLUTIONARY COMPUTATION, AND MACHINE LEARNING TO IDENTIFY TRANSPOSABLE ELEMENTS IN GENOMES

by **Wendy Cole Ashlock**

a dissertation submitted to the Faculty of Graduate Studies of York University in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

USING SIGNAL PROCESSING, EVOLUTIONARY COMPUTATION, AND MACHINE LEARNING TO IDENTIFY TRANSPOSABLE ELEMENTS IN GENOMES

by **Wendy Cole Ashlock**

By virtue of submitting this document electronically, the author certifies that this is a true electronic equivalent of the copy of the dissertation approved by York University for the award of the degree. No alteration of the content has occurred and if there are any minor variations in formatting, they are as a result of the coversion to Adobe Acrobat format (or similar software application).

Examination Committee Members:

1. Amir Asif
2. David Swayne
3. Xin Gao
4. Suprakash Datta (Supervisor)
5. Aijun An
6. Stephen Chen

Abstract

About half of the human genome consists of transposable elements (TE's), sequences that have many copies of themselves distributed throughout the genome. All genomes, from bacterial to human, contain TE's. TE's affect genome function by either creating proteins directly or affecting genome regulation. They serve as molecular fossils, giving clues to the evolutionary history of the organism. TE's are often challenging to identify because they are fragmentary or heavily mutated. In this thesis, novel features for the detection and study of TE's are developed. These features are of two types. The first type are statistical features based on the Fourier transform used to assess reading frame use. These features measure how different the reading frame use is from that of a random sequence, which reading frames the sequence is using, and the proportion of use of the active reading frames. The second type of feature, called side effect machine (SEM) features, are generated by finite state machines augmented with counters that track the number of times the state is visited. These counters then become features of the sequence. The number of possible SEM features is super-exponential in the number of states. New

methods for selecting useful feature subsets that incorporate a genetic algorithm and a novel clustering method are introduced. The features produced reveal structural characteristics of the sequences of potential interest to biologists. A detailed analysis of the genetic algorithm, its fitness functions, and its fitness landscapes is performed. The features are used, together with features used in existing exon finding algorithms, to build classifiers that distinguish TE's from other genomic sequences in humans, fruit flies, and ciliates. The classifiers achieve high accuracy ($> 85\%$) on a variety of TE classification problems. The classifiers are used to scan large genomes for TE's. In addition, the features are used to describe the TE's in the newly sequenced ciliate, *Tetrahymena thermophila* to provide information for biologists useful to them in forming hypotheses to test experimentally concerning the role of these TE's and the mechanisms that govern them.

Acknowledgements

This work was done with the support and encouragement of many people, starting with my family. My father, Richard Cole, was the person who first sparked my interest in both "junk DNA" and evolutionary computation. He and my mother, Marjorie Cole, played key roles in developing my love of learning and desire for education. My children, Charlotte, Peter, and Richard Ashlock, listened patiently as I babbled on enthusiastically about my research, learning more, I am sure, than they really wanted to know about retroviruses and ciliates. My husband, Dan Ashlock, was always available to listen to my ideas, both good and bad. In addition, I am grateful to him for inventing side effect machines, for sharing his code, and for reading and commenting on an early draft of this thesis. My friend, Justin Schonfeld, also read an early draft and made many helpful comments for which I am grateful.

Interdisciplinary work is always challenging and requires good working relationships with researchers in the other discipline. I am grateful to Ron Pearlman, professor of biology at York University, for patiently answering my questions, for explaining the

biological point of view, and for making an effort to understand my computer science techniques. I am grateful to the Pearlman lab at York University and the Fillingham lab at Ryerson University for explaining their projects to me. I thank Robert Coyne of JCVI for helping me navigate the Tetrahymena sequence data and providing helpful feedback on my work.

My thanks go to my thesis committee for all their help and support. In particular, my advisor, Suprakash Datta, who met with me weekly during my tenure as a graduate student at York. Without the discussions that occurred at those meetings and all his other help and encouragement, this thesis would never have been completed.

I thank the editors and reviewers of "The IEEE Transactions on Evolutionary Computation" and "The IEEE/ACM Transactions on Bioinformatics and Computational Biology" for their helpful comments on the papers that contributed to this thesis. I also thank the reviewers and participants at CIBCB 2010, ACM BCB 2010, and GENSIPS 2010 for their helpful comments.

Finally, I am grateful to the Natural Sciences and Engineering Research Council (NSERC), the government of Ontario, and York University for their financial support.

Table of Contents

List of Tables

List of Figures

Abbreviations

A adenine

Alu most common SINE in humans

BLAST Basic Local Alignment Search Tool

bp base pair

C cytosine

CBS chromosome breakage site

CCDS consensus coding sequences

CDS coding sequence

CpG a C base followed by a G base

DC Dissimilarity Clustering

DNA deoxyribonucleic acid

DS Dissimilarity Selection

env retroviral gene that codes for the envelope protein

ERV endogenous retrovirus

G guanine

gag retroviral gene that codes for group specific antigen

HERVd database of human endogenous retroviruses

HIV human immunodeficiency virus

HMM Hidden Markov Model

IES internal eliminated sequence

IG Information gain

IUPAC International Union of Pure and Applied Chemistry

k-mer string of length k

L1 most common LINE in humans

LINE long interspersed element

LTR long terminal repeat

MAC macronucleus

Mb megabases

MDS macronuclear destined sequence

MI mutual information

MIC micronucleus

NCBI National Center for Biotechnology Information

NCS non-coding DNA sequences

OOB out of the bag

ORF open reading frame

pam partitioning around mediods

PBS primer binding site

pol retroviral gene that codes for reverse transcriptase and integrase proteins

PPT polypurine tract

RNA ribonucleic acid

RV exogenous retroviral genomes

SEM side effect machine

SINE short interspersed element

SV support vector

SVM Support Vector Machine

T thymine

TE transposable elements

TGD Tetrahymena Genome Database

TSD target site duplication

UTR untranslated region

1 Introduction

New technologies have led to an explosion of DNA sequence data in recent years. The number of sequences in GenBank (the definitive genetic database) has grown from a few hundred in 1982 when it was founded to more than 100 million at the writing of this thesis. Generating this data was an enormous accomplishment, but it is only the beginning. Interpreting genomes is akin to debugging a computer program written in machine code for a processor for which you have no manual. The quantity of data makes manual interpretation impractical, necessitating the development of automatic techniques.

The obvious first challenge is to identify genes. Before the human genome project was completed, it was believed that this was the major challenge. But, to everyone's surprise, it turned out that less than 2% of the human genome consists of genes. Initially, the other 98% was labeled "junk DNA" and all efforts were focused on genes. Identifying genes and determining their functions is a formidable enough problem in itself. Increasingly, the importance of the so-called junk DNA (now renamed non-coding DNA) is becoming clear.

The central dogma of molecular biology is a principle about how genes work: DNA is transcribed into RNA that is translated into protein. But, there is more to running a cell than making protein. It is important when the protein is made, how much is made, when the process is stopped, which proteins are made at the same time, etc. Non-coding DNA plays an important, but not yet completely understood, regulatory role. Non-coding DNA also affects the structure of the DNA and the way it changes over evolutionary time. Because of this, it contains clues to what the genome was like in the past and its relationships to the genomes of creatures of other species as well as the relationships between two individuals of the same species. Thus, non-coding DNA affects both genome function and genome evolution.

What is stored in genetic databases are long alphabetic sequences using the alphabet {A, C, G, T}. These represent sequences of nucleotides consisting of a sugar/phosphate backbone and four types of bases: adenine (A), cytosine (C), guanine (G), and thymine (T). Genomes are assembled from shorter sequence reads (see Section 2.2.7). The sequence reads are publicly available as are many other short sequences from various research projects.

An important task towards the goal of understanding how genomes work is to label their various parts. This is called *genome annotation.* A good description of how genome annotation is performed for genes can be found in [149]. As more biological research focuses on non-coding DNA, annotations beyond gene annotations have become

increasingly important. The quality of annotations for genomes varies widely with the highest quality annotations existing for genes for those organisms which were sequenced earliest and which have been studied the most. Among the first organisms sequenced were: *Saccharomyces cerevisiae* (baker's yeast), *Caenorhabditis elegans* (roundworm), *Drosophila melanogaster* (fruit fly), *Homo sapiens* (human beings), and *Arabidopsis thaliana* (thale cress, a small flowering plant). Given the rate at which new genomes are being sequenced, it is no longer possible to take the time and care that was given to the annotation of these genomes: automation is necessary. Existing annotations can be used as starting points for annotating other sequenced genomes. The first step in annotation is to look for genes that have sequence homology with genes in already annotated genomes. Previous annotation projects are used to generate training data for the machine learning in this thesis.

The focus is on the task of annotating an important component of non-coding DNA, *transposable elements* (TEs). These are also sometimes called transposons. TEs are mobile portions of genomes. They were first discovered by Barbara McClintock [46] who called them "jumping genes." She was studying the colouration of the kernels of maize (also sometimes called Indian corn). She found that the position of TEs could block the production of pigment. The mottled pattern of the kernels is caused by the different patterns of movement of the TEs in different cells during development. TEs affect the function of genes in other organisms in a similar way. Their impact on oncogenes

(genes that cause tumours) has been a focus of study. They also can impact genome structure. For example, sometimes they carry a portion of the surrounding DNA along with them when they move. TEs create many copies of themselves within the genome, making it much longer than it would otherwise be. They can also shorten the genome. This happens when the fact that there are identical copies disrupts the DNA duplication machinery causing it to make errors. TEs can cause horizontal gene transfer (the exchange of genetic information between species). For more information about how TEs affect genome structure, function, and evolution see [97, 138, 83]. Biologists have only scratched to surface in learning about TEs in genomes.

Annotating TEs in genomes of closely related species can lead to insight into speciation, and annotating TEs in genomes of individuals of the same species can lead both to insight into how TEs work and insight into individual genetic differences and genetic diseases. Also of interest to biologists is study of the organism's policing methods for removing or silencing unwanted insertions into their genomes.

All genomes, from bacterial to human, contain TEs. Primate genomes are about half TEs; other mammalian genomes are about one-third TEs. Some genomes have less (for example insect genomes are about 20% TEs) and others have more (plants – more than 60%). Although TEs originate from viral insertions, and thus have genes, they are classified as non-coding DNA and are hard to identify because they are often fragmentary or heavily mutated, making it difficult to pinpoint characteristic genes or

other characteristic subsequences. They evolve quickly, meaning that they have little sequence identity with each other, making it difficult to identify them using sequence homology. Identifying even fragments of TEs is useful because TEs can share genes with each other, meaning that a fragmentary TE is often functional. Despite their importance and abundance, few reference genomes have good (meaning mostly correct and nearly complete) annotations.

In this thesis, novel features for the detection of TEs, for distinguishing different types of TEs, and for learning about the character and function of TEs are developed. These features are of two types. The first type are based on signal processing techniques and include measures of randomness and ways of detecting the distinctive reading frame structure of retroviruses. The second type of feature uses a computational intelligence technique (genetic algorithm) to discover unique qualities of different types of TEs. These features are used to classify different types of TEs and to scan the human and fruit fly genomes for TEs. In addition, these techniques are applied to the study of TEs in a recently sequenced organism with unique characteristics, *Tetrahymena thermophila*. Use of these features provides insight into its unique and little understood TEs.

1.1 Approach

This thesis presents novels methods for generating DNA sequence features. Millions of potential features are generated and then a selection process extracts those that are

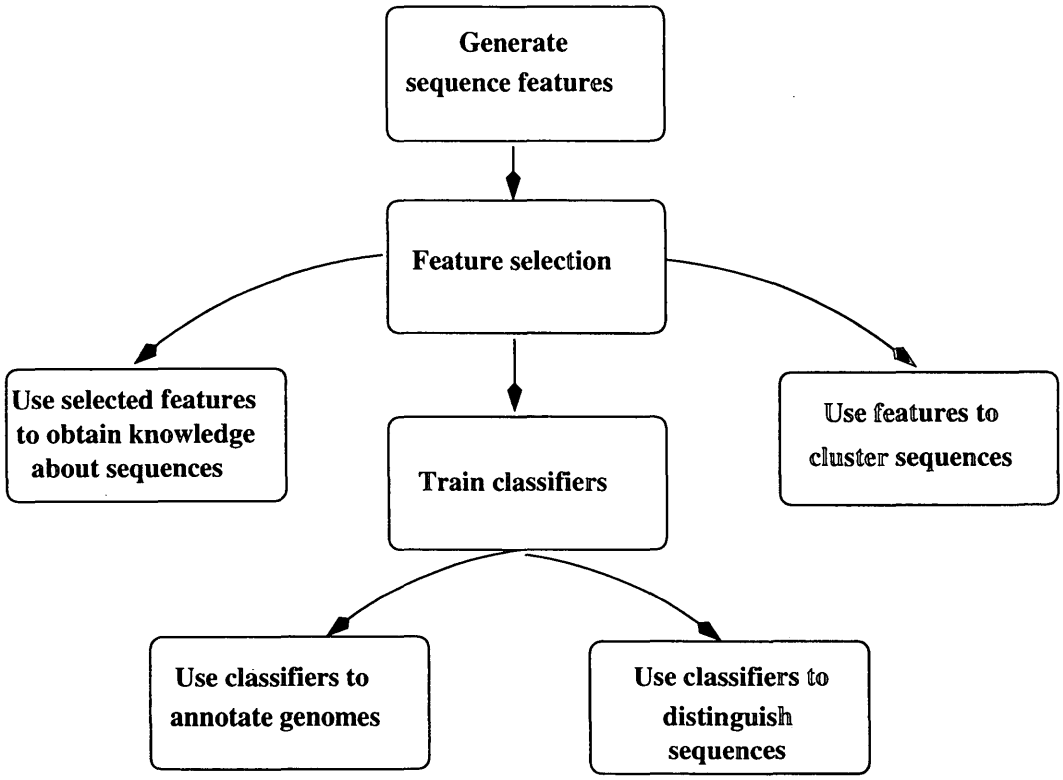


Figure 1.1: Approach used in this thesis for improving understanding of TEs in genomes.

interesting and comprehensible for various problems involving identifying TEs and distinguishing different types of TEs. These features are used to annotate genomes, classify sequences, and also to provide descriptions of the sequences intended to inspire biologists to form hypotheses for experimentation. This process is summarized in Figure 1.1.

1.1.1 Features Based On Signal Processing

Useful sequence features can be developed based on statistical properties inherent in particular regions of the genome. For example, when DNA codes for proteins, it uses a genetic code (see Table 1.1) consisting of groups of three bases (codons). Each codon codes for an amino acid or a start or stop signal. (In the table, amino acids are represented by their one-letter abbreviation.) Strings of amino acids make up proteins. Most of life is thought to use a standard genetic code, but a few variant codes are also used. (*Tetrahymena* was the first organism discovered to use a variant code in its nuclear genome.) The genetic code is degenerate – multiple codons code for the same amino acid since, with four nucleotides, there are 64 combinations possible to code for only twenty amino acids. This degeneracy leads to detectable statistical properties in the genome or regions of the genome. For example, in regions that use this genetic code (protein coding regions), the third position of the codon is more variable than the other two positions. In addition, sometimes a particular choice of codon for a given amino acid is preferred. This "codon usage" bias can be detected with statistical techniques.

Regions that do not use the genetic code also often have statistical features that can be detected. Some regions have biases for particular bases. This is characterized in terms of AT-richness (percentage of bases which are A or T). A C base directly followed by a G base is written CpG. These pairs are rare in most of the genome because their

chemistry encourages mutation. There are, however, regions of the genome where they are common. These regions are called CpG islands and can be detected statistically. It is also common to have short sequences repeated many times. These are called tandem repeats and are also statistically detectable.

These bioinformatic approaches inspired the development of the features described in Chapter 3. Converting the sequences into numeric values enables the use of signal processing techniques designed to discover periodicities in time series. This allows the incorporation of biological knowledge of the sequence structure into useful features.

1.1.2 Side Effect Machines

Another approach to developing useful features results in those described in Chapter 4. This approach uses a genetic algorithm to evolve finite state machines augmented with one counter per state, called side effect machines (SEMs), that produce sequence features. Side effect machine features are the values of the counters after running the sequence through the machine, normalized by sequence length. The counter values are side effects of running the string through the finite state machine. After the SEM is created, its features can be used with a classifier, like a support vector machine (Section 6.27) or a random forest (Section 6.1.2). SEMs were introduced in [13] and [10] where they were used for classifying synthetic and biological DNA strings. They have also been used in [10, 12, 11, 14, 117, 28, 9, 17, 96, 18].

	T		C		A		G	
T	TTT	F	TCT	S	TAT	Y	TGT	C
	TTC	F	TCC	S	TAC	Y	TGC	C
	TTA	L	TCA	S	TAA	STOP	TGA	STOP
	TTG	L	TCG	S	TAG	STOP	TGG	W
C	CTT	L	CCT	P	CAT	H	CGT	R
	CTC	L	CCC	P	CAC	H	CGC	R
	CTA	L	CCA	P	CAA	Q	CGA	R
	CTG	L	CCG	P	CAG	Q	CGG	R
A	ATT	I	ACT	T	AAT	N	AGT	S
	ATC	I	ACC	T	AAC	N	AGC	S
	ATA	I	ACA	T	AAA	K	AGA	R
	ATG	START	ACG	T	AAG	K	AGG	R
G	GTT	V	GCT	A	GAT	D	GGT	G
	GTC	V	GCC	A	GAC	D	GGC	G
	GTA	V	GCA	A	GAA	E	GGA	G
	GTG	V	GCG	A	GAG	E	GGG	G

Abbreviation	amino acid	Abbreviation	amino acid
A	Alanine	L	Leucine
R	Arginine	K	Lysine
N	Asparagine	M	Methionine
D	Aspartate	F	Phenylalanine
C	Cysteine	P	Proline
Q	Glutamine	S	Serine
E	Glutamate	T	Threonine
G	Glycine	W	Tryptophan
H	Histidine	Y	Tyrosine
I	Isoleucine	V	Valine

Table 1.1: Standard Genetic Code

The finite state machines have n states each with m transitions. Each transition corresponds to a member of the string alphabet. So, for example, a SEM operating on a DNA string using the alphabet {A,C,G,T}, as in [13, 10], has four transitions for each state. When a string is fed through the machine, a count is kept for each state of how many times the string passes through that state. After the entire string has been run through, these counts are normalized by dividing by the string length. These n counts constitute a vector in $\mathbb{R}^n$.

SEMs are selected using a genetic algorithm. The genetic algorithm evolves a population of SEMs for a particular set of training data, evaluating fitness based on how well each SEMs feature set performs in a classifier trained and tested using different portions of the training data. The most fit member of the population is chosen to generate the final feature set. The genetic algorithm can be run many times to produce different SEMs and thus many sets of features.

1.2 Types Of Sequences

Figure 1.2 shows the different types of TEs. This thesis focuses on four important types of TEs: LTR retrotransposons, solitary LTRs, short interspersed elements (SINEs), and internal eliminated sequences (IESs). IESs are thought to be degraded DNA transposons. The first three were chosen because of their importance in the human genome and the last because of the superior quality of the data due to the unique properties of ciliate

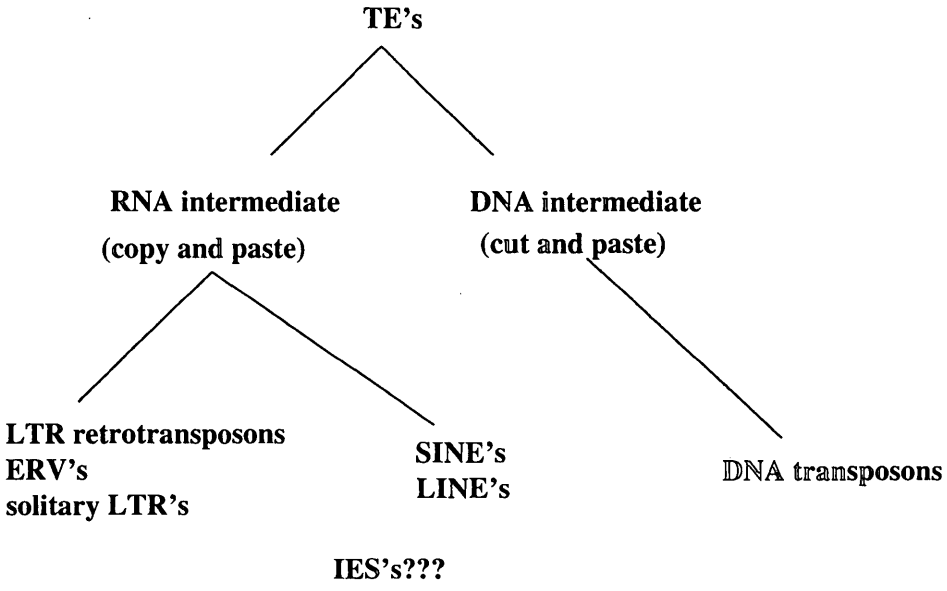


Figure 1.2: Types of TEs.

genomes. Retrotransposons are TEs that reproduce via an RNA intermediate, i.e., their DNA is copied to RNA that is then used to create a DNA copy that is inserted in a new place in the genome. Autonomous retrotransposons (those that encode reverse transcriptase, the protein guiding the RNA to DNA copying process) include LTR retrotransposons and endogenous retroviruses (ERVs). These elements have a repeated sequence, called a long terminal repeat (LTR) at their ends. They play a role in gene expression, the creation of new genes, the arrangement of genes in the genome, and genetic diversity within a species. Solitary LTRs form when the internal region is deleted due to homologous recombination of the matching LTRs at either end of the retrotransposon [58, 140]. They are more abundant than complete LTR retrotransposons in the human genome.

Since solitary LTRs contain regulatory elements, they can have a functional role in the genome. Identifying solitary LTRs is important to biologists interested in the evolutionary history of LTR retrotransposon insertions [85], in their role in health and disease [31, 37, 80, 112], in their role in gene function [128, 31, 103], in genomic mechanisms for suppressing their expression [37], and in their role in evolution and speciation [80]. SINEs have many characteristics in common with solitary LTRs and, thus, are easily confused with them. SINEs are the most common TE in the human genome, making up about 11% of it. Since insertions of TEs are often disruptive to the organism, all organisms have some method of deleting at least some of them. Ciliates, a type of protozoan, have a unique way of excising their TEs: they have two genomes, one with (the micronuclear or MIC genome) and one without TEs (the macronuclear or MAC genome). Both the MAC and MIC genomes of the ciliate *Tetrahymena thermophila* has recently been sequenced, enabling study of these TEs (called internal eliminated sequences or IES). Comparison of the two genomes allows nearly exact identification of IESs.

1.2.1 Genes

Dectection of genes is complicated by the fact that they contain both regions that code for proteins, called exons, and regions that do not, called introns and UTRs (untranslated regions). Figure 1.3 illustrates this. Genes function by first being transcribed from DNA into RNA and then being translated into protein. The UTRs occur at the beginning and

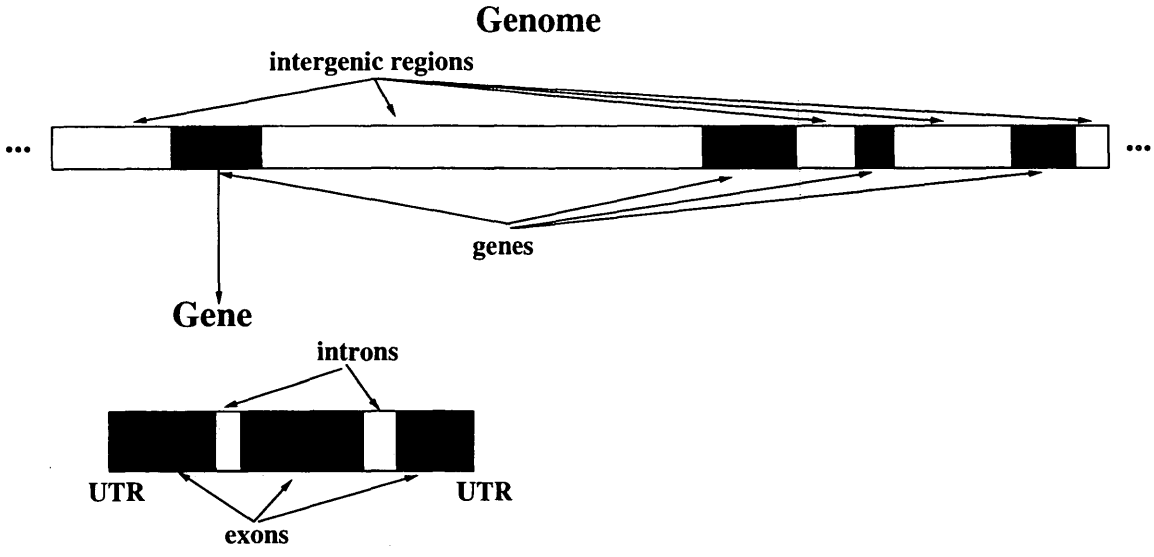


Figure 1.3: Structure of genome and gene.

end of the gene, and, as their name suggests are not translated into protein. The introns also are not translated into protein. They are transcribed into RNA, but they are spliced out of the RNA transcript prior to translation.

1.2.2 Retroviruses

Many TEs originate from viral insertion, so viral sequences are included in the study, and, in particular, retroviral sequences. Viruses are genetic parasites that can only replicate using the cellular systems of a host. They are similar to living organisms in many ways: they can die; they evolve by natural selection, and virus species can become extinct. The most studied viruses are associated with disease (examples include the H1N1 influenza virus, the herpes virus, and the HIV virus), but the majority of known viruses do not cause

disease, and some are even beneficial to their hosts. Viruses infect cells from all types of life. There are viruses that infect animals, plants, bacteria, fungi, algae, even other viruses. Some viruses are species specific; they only infect cells from a particular species. Many others invade cells from a broad range of species. Viruses are ancient. They have been part of life and part of evolution for hundreds of millions of years. Virus particles are simple. They have only two or three parts: genes made of RNA or DNA, a protein coat protecting the genes, and, sometimes, an envelope made of lipids surrounding the entire particle.

Viruses have two possible life strategies: some are acute, and some are persistent. Acute viruses kill their host cells. Persistent viruses integrate into their host cells, becoming a permanent part of them. Retroviruses are persistent RNA viruses. When they enter a cell, their RNA is converted to DNA and inserted into the DNA of their host. New viruses are created using the host cell's transcription and replication machinery. These are the viruses of interest in this thesis. Retroviruses encode reverse transcriptase, an enzyme that converts RNA into DNA (backwards from the RNA to DNA process used with genes), so they can copy and paste their sequences in many locations in the genome of the host cell. Retroviruses are studied because of their role in diseases like HIV, and their potential for use in gene therapy. For more information about viruses and their role in evolution, see [138].

1.2.3 Endogenous Retroviruses

When a retrovirus inserts into a germ cell (a sperm or egg cell), the retroviral DNA is inherited. This DNA is an endogenous retrovirus (ERV). Viruses are particular as to the type of cell they can insert into, so not all retroviruses can insert into germ cells. HIV, for example, can insert only into cells in the immune system and cells in the central nervous system. Thus, HIV is unlikely to become an ERV. RNA retroviruses are referred to as exogenous or wild viruses to distinguish them from ERVs. ERVs are also sometimes referred to as proviruses, and sequences that appear to derive from retroviral insertions but are not related to a known retrovirus are called LTR retrotransposons. Human ERVs are referred to as HERVs, cow ERVs as BERVs (bovine ERVs), sheep ERVs as OERVs (ovine ERVs), etc.

In addition to the original insertion, copies of the retrovirus are made and inserted elsewhere in the genome. It is estimated that about 8% of the human genome is made up of ERVs [144]. The ERV will, of course, only become a permanent feature in the species's population if it is not harmful to the host, or, at least, not too harmful. If it is beneficial, it will undergo positive selection. Some ERVs become defective or non-functional due to mutation. Some are found in similar locations in distantly related species, implying they have been part of the genome for a very long time. A retrovirus named Phoenix was estimated to have been part of the human genome for five million

years [41]. To put this in perspective: modern humans have only existed for 50,000 - 100,000 years. Phoenix was shown to be still capable of producing infectious particles. ERVs like Phoenix are useful to evolutionary biologists as living molecular fossils which help them determine the relationships among species.

Although the complete biological significance of ERVs is still not fully understood, they have been shown to be important in many ways. They affect the structure of our DNA; they are associated with cancer and other diseases; they perform useful functions like producing the protein which causes immunosuppression in the human placenta; and, they may protect us from infection by exogenous retroviruses. Our immune system appears to have arisen from a retrovirus, although the viral ancestor of adaptive immunity remains to be found [138]. In addition, ERVs can be found in the same locations in the chromosomes of related species. Information about when the species diverged can be gleaned from calculating the time since insertion based on the number of mutations. Some retroviruses integrated into the genome hundreds of millions of years ago. For reviews, see [23, 84, 136, 60, 50].

In addition to the original insertion, copies of transposons are made and inserted elsewhere in the genome (retrotransposition). This is why they comprise so much of the genome. Some retroviruses are repeated only a few times; others are repeated hundreds of times [84]. Sometimes these copies carry part of the cellular DNA with them, rearranging the genome. Sometimes they affect the function of neighbouring genes just

through their presence. For example, they can interfere with regulatory sequences. Retroviruses can cause cancer in this way [78]. An example is feline leukaemia virus which causes cancer by activating an inactive cancer-causing gene (oncogene) in the cell. Some cancer-causing retroviruses carry oncogenes. The Rous sarcoma virus is an example of this [145].

1.2.4 Solitary LTRs

Solitary LTRs are common in the human genome. We know they are functional because they are conserved and transcribed [60, 102, 95], but their function is not well understood. They are related to promoter regions for genes. Because they insert copies of themselves in multiple locations in the genome, they impact genome size and structure. Identifying families of solitary LTRs is important to the study of genome evolution as they serve as biological markers.

A diagram of an LTR retrotransposon can be seen in Figure 1.4a; a solitary LTR in Figure 1.4b. A solitary LTR is created when the internal region of the LTR retrotransposon is deleted through homologous recombination [58, 140] as shown in Figure 1.6. This means that the two nearly identical LTR sequences bind together causing the region between them to be deleted.

The basic solitary LTR structure consists of regions copied from three different parts of the retroviral RNA genome: the R region, the U3 region, and part of the U5 region.

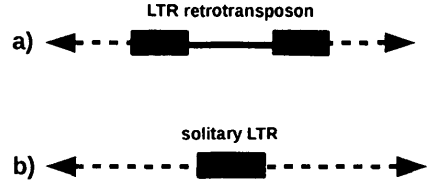


Figure 1.4: LTR retrotransposon and solitary LTR – rectangles represent LTRs; solid line represents viral genes; dotted line represents genomic DNA.

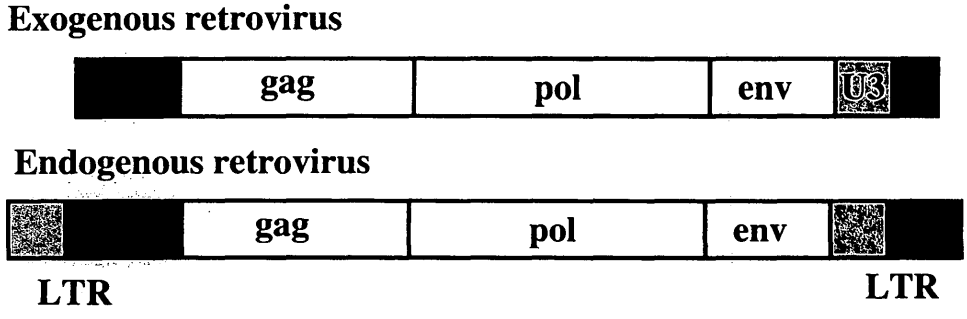


Figure 1.5: Comparison of structure of exogenous and endogenous retroviruses. The three genes, gag, pol, and env, are labeled, as are the regions of the exogenous retrovirus that make up the LTR of the endogenous retrovirus.

These regions are shown in the diagram in Figure 1.5. Solitary LTRs rarely contain coding segments.

The problem of identifying solitary LTRs is more challenging than that of identifying LTR retrotransposons, because many of the methods for identifying LTR retrotransposons rely on the existence of the LTRs at either end of the sequence. However, solitary LTRs are more common than complete LTR retrotransposons and more likely to be transcribed [102]. An LTR retrotransposon called HERV-K(HML-2) is estimated to have

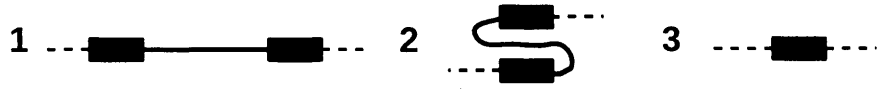


Figure 1.6: Formation of solitary LTR. 1 shows the original LTR retrotransposon; 2 shows homologous recombination; 3 shows the resulting solitary LTR.

ten times as many solitary LTRs as complete copies in the human genome [126]. All techniques that rely on LTRs occurring in pairs fail to find solitary LTRs.

1.2.5 Non-LTR Retrotransposons

Like ERVs, SINEs (short interspersed elements) are a type of TE that use reverse transcriptase to transcribe. Sequences with this property are called retrotransposons. They are members of a different class of retrotransposon than LTR retrotransposons, usually referred to as non-LTR-retrotransposons. SINEs are related to another type of retrotransposon, called a LINE (long interspersed elements). LINEs range from 900-6000 base pairs (bps); SINEs from 200-400 bps. LINEs have a gene for reverse transcriptase, but SINEs do not. It is believed that SINEs coopt the LINE gene in order to make copies of themselves. SINEs and LINEs are derived from RNA polymerase transcripts. RNA polymerase is the enzyme that catalyses RNA synthesis from DNA. LINEs make up about 21% of the human genome; SINEs about 11% [104]. The most common transposon in humans is the SINE family Alu. Our genome has about 300,000 copies of Alu, one for every 6K of DNA. LINEs and SINEs differ from ERVs in that they do not have long

terminal repeats (LTRs), identical sequences at their beginnings and ends.

SINEs have a trinary structure and contain no coding segments. They consist of a head, a body, and a tail. They are GC rich and rich in CG dinucleonides and have A rich tails and T rich heads. Their tails often consist of repeated sequences of length 1-8 bp. They often have poly-A tails, and their RNA transcripts have a conserved hairpin loop secondary structure [127]. See [42] for more information about SINEs.

1.2.6 Tetrahymena TEs

Tetrahymena thermophila is a single-celled animal that has been much studied as a model organism. Research on it has led to Nobel prize winning discoveries about telomeres and catalytic RNA. It has many genes in common with human beings and is easily cultured in the laboratory making it important in medical research. It is a type of protozoan called a ciliate, so called because it moves around using hairlike structures called cilia.

Ciliates have the unique and interesting property that they have two nuclei, called the micronucleus (MIC)and the macronucleus (MAC). The MAC, though larger than the MIC, actually has less genetic information. It is larger because it has many copies of all its genetic material (i.e., it is polyploid). The MIC genome, like the human genome, is diploid with only two copies of all its genetic material. All the genetic information in the MAC is also contained in the MIC. Sequences that occur in both the MIC and the MAC are referred to as MDSs (macronuclear destined sequences). The MIC is used by

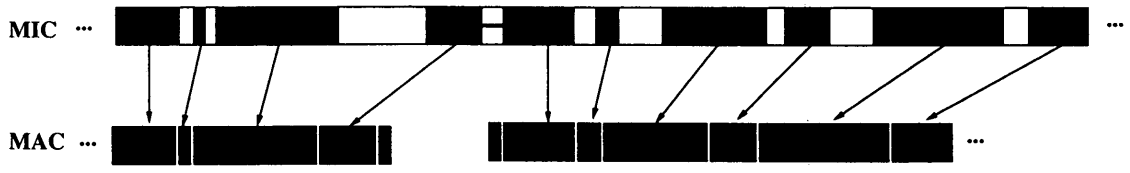


Figure 1.7: Relationship between the MIC and MAC genomes. MDSs are represented by filled blue rectangles and IESs by unfilled rectangles. The horizontal red line represents a chromosome breakage site and the shaded green rectangles represent the telomeres that are added to the ends of each chromosome-like sequence in the MAC.

the organism solely for sexual reproduction. The MAC is used to run the cell. During the sexual phase of the life cycle, a new MAC develops from a diploid MIC zygotic fertilization product, and thousands of internal eliminated sequences (IESs) are removed from the developing MAC. These MIC-limited IESs comprise over 30% of the MIC genome.

The relationship between the sequences in the MIC and MAC are shown in Figure 1.7. The MIC has five diploid (two copies) chromosomes. The MAC has hundreds of chromosome-like pseudomolecules with about 45 copies of each. Short sequences called chromosome breakage sites (CBS) in the MIC indicate where to create a new chromosome-like pseudomolecule in the developing MAC. Such a sequence is represented with a red line in Figure 1.7. When a breakage occurs, telemeric repeats (repeats of the sequence GGGGTT) are added to the ends of the new molecule. For more information, see [39, 34].

The complete MAC genome sequence of Tetrahymena thermophila is available[1] and

[1] http://www.ciliate.org

the MIC sequence was recently completed[2]. It is believed that IESs are remnants of DNA transposon insertions. DNA transposons are transposons that do not transcribe using reverse transcriptase; they copy using a cut-and-paste rather than a copy-and-paste mechanism. They are common in bacteria and protists; less common but still present in eukaryotic organisms including humans. The IES elimination pathway is similar to the silencing of transposons in metazoans, just more extreme, as they are entirely eliminated, not just silenced.

Having the sequences of both the MIC [27] and MAC [45] of Tetrahymena thermophila enables detailed bioinformatic study of the IESs. Formerly, their study was based on just a handful of identified sequences. This thesis uses a data set of nearly 6000 IESs assembled by comparing the MIC genome with the MAC genome and extracting the sequences that exist only in the MAC (Algorithm 1).

This is the first time this has been done for this organism. The techniques developed for the study of retroviruses are used to bioinformatically describe these sequences. Experimental biologists will be able to use this work to form hypotheses and perform experiments to further the study of this fascinating and important organism.

Since TEs in ciliates can be identified with near certainty, they provide a unique opportunity to study structural characteristics of TEs. In addition, since ciliates share many genes with humans, there is substantial value in gaining an understanding of the

[2]http://www.broadinstitute.org/annotation/genome/Tetrahymena/MultiHome.html

Algorithm 1: Find IESs using BLAST

Data: BLAST database of m MAC scaffolds MAC, MIC contigs MIC, minimum identity $minI$, maximum gap percentage $maxgap$, minimum match length $minlen$, minimum length of group of matches $minmatch$

Result: IES sequences

for *each contig in MIC* **do**
- BLAST against MAC using blastn with megablast, no filtering, and $e < 0.001$;
- **for** *all BLAST hits* **do**
  - Store sums of lengths of hits for each MAC scaffold in each orientation in array *hitlen*
- **end**
- Sort *hitlen*;
- **for** $i \leftarrow 1$ *to* $m * 2$ **do**
  - **if** $hitlen[i] > minmatch$ **then**
    - Extract hits for scaf i that have identity $> minI$, gaps $< maxgap$, length $> minlen$;
    - Arrange in order for both MIC and MAC;
    - Trim ends so there is no gap bigger than 30000 and get rid of overlaps;
    - Extract IESs (gaps in the match);
  - **end**
- **end**

end

return IES sequences;

mechanisms involved in such things as IES excision. These mechanisms are likely to be basic to other processes occurring in "higher" eukaryotes.

1.3 Resources

Training data from several sources is used. Consensus sequences for TEs from a few dozen eukaryotic organisms (i.e. non-bacterial organisms) have been compiled in a database called RepBase [65]. Consensus sequences are built from multiple examples of a sequence. The number of examples can range from several to hundreds. The sequences are aligned (i.e., their corresponding bases are matched, with some sequences having insertions or deletions not in the other sequences, referred to as gaps). A new sequence, the consensus sequence, is built by taking a majority vote for the base in each position of the sequence. This is an inexact process. Judgement is involved in deciding that two sequences are the "same." Aligning the sequences quickly and correctly is an area of active research. The majority vote process is supposed to filter out mutations, but it could also be filtering out important sequence information. There may be no actual sequences that are exactly like a given consensus sequence. In Section 5.2.3 some problems with using consensus sequences for sequence analysis are discussed. However, they are generally considered to be useful and representative of a particular type of sequence. Some examples of sequences represented by their consensus sequence in RepBase are: the Harlequin LTR retrotransposon in humans, the MERV1 ERV in mice, and the F524

SINE in rice.

The organisms with sequences in RepBase include everything from phytoplankton to green plants to insects to domestic animals to humans. At the time of this writing, there are 31,022 sequences in RepBase. These include 14,568 LTR retrotransposons, 4098 ERVs, and 641 SINEs. Also included are DNA transposons (7248), non-LTR retrotransposons other than SINEs (3807), simple repeats (515), pseudogenes (117), and integrated virusus (28). There are no IESs in RepBase or any sequences from ciliates, and although RepBase contains sequences for DNA transposons, which are believed to be the type of TEs from which IESs originated, previous researchers have been unable to find any homology between *Tetrahymena thermophila* IESs and the sequences in RepBase. RepBase has been used to create partial annotations in many genomes using a program called RepeatMasker [121]. RepeatMasker uses homology with sequences in RepBase to identify fragments of TEs in sequenced genomes. Some TEs are represented by many fragments.

In addition to these general purpose databases, there are organism specific databases. An organism with good annotations is *Drosophila melanogaster*. These annotations were created using a variety of bioinformatics tools together with hand annotations [107]. The human genome is complex and difficult to annotate, but much attention and funding has been given to its study, resulting in annotations generated from multiple sources without much coordination or oversight. The dual genome structure of *Tetrahymena thermophila* makes identifying IESs a matter of comparing the two genomes using BLAST and ex-

tracting the sequences that exist in the MIC but not in the MAC. Due to the smallness of the ciliate research community and the newness of the sequencing, other annotations to the genome (like where the genes are) are incomplete and in a state of constant update. This work concentrates on sequences from these three organisms.

Two sources used for identifying ERVs in the *Homo sapiens* genome are RetroSearch [139] and Retrotector [124]. RetroSearch uses a method similar to RepeatMasker's with an additional step to string together the fragments. Retrotector scans the genome for "motifs," structural features of various kinds, and then assembles complete ERVs. Both of these approaches result in annotations that are very likely to be correct, but are also likely to be incomplete. The LTRs in the RetroTector sequences are annotated, enabling their use for generating training data for solitary LTRs as well.

For fruit fly sequences, the annotated genome from FlyBase [134] is used. For Tetrahymena IES and MDS sequences, the sequences are generated using BLAST (Basic Local Alignment Search Tool) [6] with the sequenced MIC and MAC geneomes. BLAST is a heuristic algorithm for determining whether sequences are similar. It is widely used by biologists.

1.4 Thesis Organization

This thesis starts with a review of related work (Chapter 2). Chapter 3 presents statistical features developed for sequence classification and identification, and Chapter 4 presents

side effect machine features. Chapter 5 presents an extensive analysis of the SEM fitness landscape and demonstrates how feature selection can be used to find features that provide biological insight. Chapter 6 presents the results of classification using both sorts of features. The use of these features in a scanner for TEs is described in Chapter 7. Finally, a discussion of how the features presented in this thesis can be used to do unsupervised learning is provided in Chapter 8. This demonstrates how SEM features can be used to learn about the sequences and help biologists formulate hypotheses for future experimentation. Much of the material in this thesis has been published in two journal papers, [17] and [18], and three conference papers [14, 16, 15].

2 Related Work

As an interdisciplinary thesis, this thesis makes contributions both to computer science and biology. The computer science contribution is in developing bioinformatic techniques for DNA sequence analysis. The biological contribution is in applying these techniques to the problem of detecting and classifying TEs. This chapter summarizes the work done by previous researchers from these two areas.

2.1 Related Work For Analyzing DNA Sequences Using Signal Processing And Machine Learning

Digital signal processing techniques have been applied to various problems in genomics. One of the most important is gene finding [89, 73, 118]. Before the Human Genome Project was completed in 2003, it was believed that chromosomes were strings of genes. In fact, it turns out that only a small percentage of the human genome (about 2%) consists of genes. In addition, a gene is not a straightforward sequence beginning with a start code and ending with a stop code. Instead, it is a complex mixture of regions that code for protein (exons) and regions containing regulatory and other elements (introns). Thus, the

problem of identifying genes and the exons within them is a non-trivial one. Overviews of how signal processing has been used in gene finding and in other genomics problems can be found in [82, 137, 7, 4].

2.1.1 Numerical Representations Of DNA Sequences

In order to apply signal processing techniques to DNA sequences, it is necessary to turn them into numeric sequences. There are many ways to do this. One common way is to use binary indicator sequences: four sequences, one for each nucleotide: A, C, G, or T. The A-sequence, for example, would have a 1 everywhere the sequence had an A and a zero elsewhere. This method was first used by Voss [141] and is referred to as the Voss representation. It has since been used by many others including [131, 40, 72].

There is some concern that results based on these sequences could be an artifact of the representation. In [113], Rushdi and Tuqan compare the Voss representation to four other numeric representations and show that they all yield the same DNA Fourier spectrum (explained in Section 2.1.2). The four representations they examine are: tetrahedral mappings, quaternions, simplex mappings, and Z-curve mappings.

Tetrahedral mappings map $\{A, C, G, T\}$ onto $\{1+j, -1+j, -1-j, 1-j\}$, where $j = \sqrt{-1}$. These values represent the corners of a tetrahedron projected onto the complex plane. Quaternions, a generalization of the tetrahedral mapping, map $\{A, C, G, T\} \rightarrow \{i+j+k, i-j-k, -i-j+k, -i+j-k\}$ where these are hypercomplex numbers such

that $i^2 = j^2 = k^2 = ijk = -1$. Simplex mappings are a transformation of tetrahedral mappings from four sequences to three sequences.

Z-curve mappings create three sequences based on pairings of bases: AG/CT, AC/GT, and AT/GC. These pairings are used because they have biological meaning: AG/CT distinguish purines and pyrimidines; AC/GT distinguish amino and keto bases; AT/GC distinguish bases with weak and strong hydrogen bonds. The sequence has a 1 if the base is one of the first pair, -1 if the base is one of the second pair. The name is derived from Z-curves [152], a method used to create a graphical representation of DNA.

Wang and Schonfeld further develop the theory needed for comparing representations in [143]. They use their theory to compare the Voss representation to the representation that uses two sequences such that in one sequence there is a -1 for A, a 1 for T, and zeros for C and G, and in the other sequence there is a -1 for C, a 1 for G, and zeros for A and T. They show that these different representations do *not* produce equivalent results. They also compare the Voss representation to a mapping that creates four sequences: one sequence maps A to $\frac{1}{\sqrt{2}}$ and G to $-\frac{1}{\sqrt{2}}$ and C and T to 0; one maps T to $\frac{1}{\sqrt{2}}$ and C to $-\frac{1}{\sqrt{2}}$ and A and G to 0; one maps A and G to $\frac{1}{\sqrt{2}}$ and C and T to zero; the last maps C and T to $\frac{1}{\sqrt{2}}$ and A and G to zero. They show that this representation produces the same Fourier spectral results as the Voss representation. They show that rotation is the unique equivalent transformation from one mapping to another that leads to consistent results, and that, when there is inconsistency, it increases as the window size of the analyzed

DNA sequences increases.

Real number mappings are used as well, particularly with AR models (see Section 2.1.3). A common one is $A = -1.5; T = 1.5; C = 0.5; G = -0.5$. This is used, for example, in [33]. This mapping has the nice property that it is easy to calculate the sequence on the opposite strand: just multiply by -1 and reverse the sequence. For a comparison of the use of different mappings on a particular problem, see [109].

2.1.2 Discrete Fourier Transform

The discrete Fourier transform (DFT) of a sequence X[n] of length N is defined as:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} X[n] e^{-\frac{j2\pi kn}{N}} \tag{2.1}$$

It is known to be useful in finding periodicities, so this was its first application in genomics. In [132], Trifonov found periodicities of 3, 10.5, 200, and 400 bases. He explained the 10.5-periodicity based on the need for the DNA to deform and fold in the nucleus and based on the coiled structure of some of the proteins coded for by the DNA. The 200- and 400-periodicities were explained by the segmented organization of the genome. The 3-periodicity was found only in protein coding regions (exons), which led to the use of the DFT in gene finding.

The reason for the 3-periodicity in protein coding regions is that the identity of the third base of a codon matters less than the identities of the other two bases due to the way the genetic code is constructed. This generates a 3-periodicity in the DFT, making the

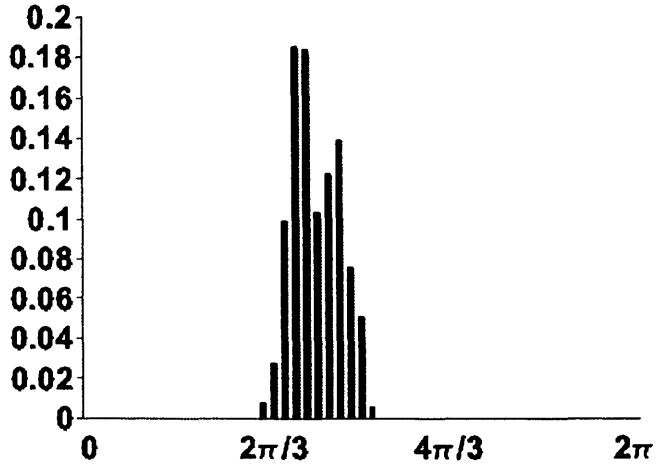


Figure 2.1: Histogram of phase values computed with a sliding window on a sequence from a coding region from the human genome.

value of the DFT at $\frac{N}{3}$ particularly useful for analyzing DNA sequences. The DFT produces a complex number that has a magnitude and a phase (r and θ in polar coordinates). Both the magnitude and phase have been used to distinguish between protein coding and non-protein coding regions of the genome [40, 72, 133]. High magnitudes at $\frac{N}{3}$ signify coding regions.

The phase value is used in [72] by Kotlar as part of his so-called Spectral Rotation Measure. This measure relies on the fact that histograms of phase values computed for a sliding window on a region of a genome look considerably different depending on whether the region is protein coding or not. Figure 2.1 shows an example histogram for a coding region of the human genome; Figure 2.2 shows a histogram for a non-coding region. These phase values were calculated using a 240 bp sliding window on the DNA sequence, sliding 3 bp between calculations.

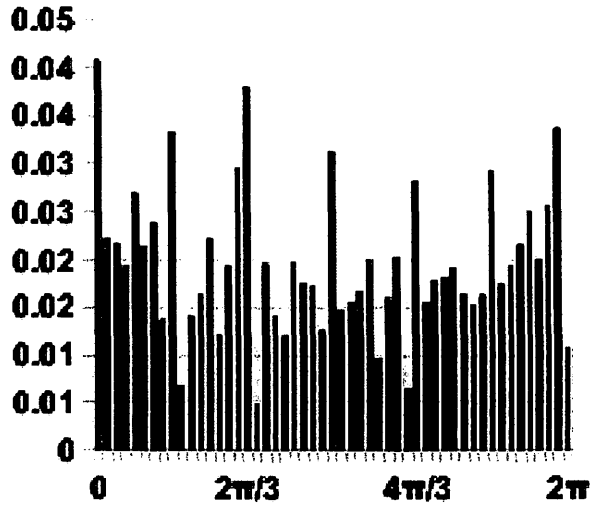


Figure 2.2: Histogram of phase values computed with a sliding window on a sequence from a non-coding region from the human genome (reprinted from [16]).

Kotlar's spectral rotation measure is given by:

$$|V|^2 = \left| \frac{e^{-i\mu_A}}{\sigma_A} A(s) + \frac{e^{-i\mu_T}}{\sigma_T} T(s) + \frac{e^{-i\mu_C}}{\sigma_C} C(s) + \frac{e^{-i\mu_G}}{\sigma_G} G(s) \right|^2 \tag{2.2}$$

where $A(s), T(s), C(s)$, and $G(s)$ are complex numbers representing the values of the DFT at frequency one-third for the Voss representation of DNA sequence s; μ_A, μ_T, μ_C, and μ_G are the approximated average phase values for coding regions, and $\sigma_A, \sigma_T, \sigma_C$, and σ_G are the standard deviations of the phases for coding regions. The μ and σ values are species specific. This measure has higher value for coding regions than for non-coding regions, because it selects out the parts of $A(s), T(s), C(s)$, and $G(s)$ pointing in the direction of the peak value of the histogram similar to that shown in Figure 2.1. Kotlar also defines a G Rotation Measure based only on the binary sequence defined by

G bases:

$$|V_G|^2 = |e^{-i\tilde{\mu}_G} G(s) + |G(s)||^2 \tag{2.3}$$

where $\tilde{\mu}$ is the value of $\{\mu, \mu + \frac{2\pi}{3}, \mu - \frac{2\pi}{3}\}$ which is maximal (an adjustment for reading frame). Kotlar finds that both the Spectral Rotation Measure and the G Rotation Measure are effective for finding coding regions in yeast, and, in fact, perform similarly.

In addition to being useful for detecting coding regions, the phase histogram gives information about which reading frame is being used (which is why Kotlar needs to make the $\tilde{\mu}$ adjustment). An insertion or deletion (i.e., a shift in reading frame) in exons in coding regions shifts the reading frame by $-\frac{2\pi}{3}$ and $\frac{2\pi}{3}$ respectively. Figure 2.3 shows the impact of deleting one base from the middle of the sequence generating the histogram in Figure 2.1. The two reading frames in the sequence are represented as two groups in the histogram shifted $\frac{2\pi}{3}$ from each other. Similarly, Figure 2.4 shows the three reading frames created when a second base is deleted.

Fourier magnitude and phase values at frequency one-third are quick and easy to calculate. In [90], the following formula is derived from the position count functions, C_i, where $i \in 1, 2, 3$. The value of C_i is the number of ones in the ith position of each group of three scanning across the sequence. The Fourier phase value at frequency 1/3 is:

$$\arctan\left(\frac{\sqrt{3}(C_2 - C_1)}{2C_0 - C_1 - C_2}\right). \tag{2.4}$$

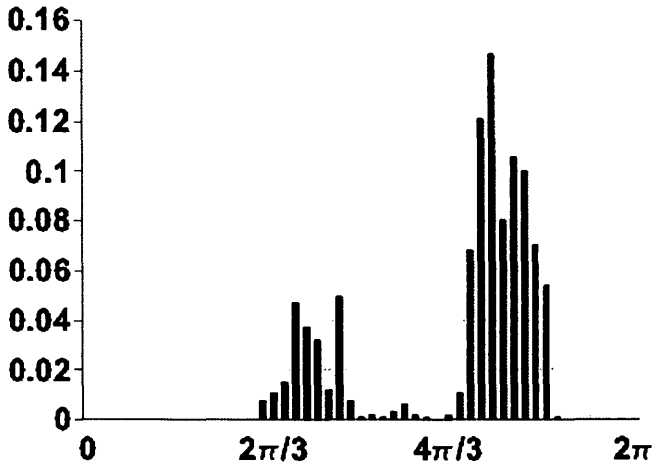


Figure 2.3: Histogram of phase values computed with a sliding window on the sequence from Figure 2.1 with one base deleted creating two reading frames.

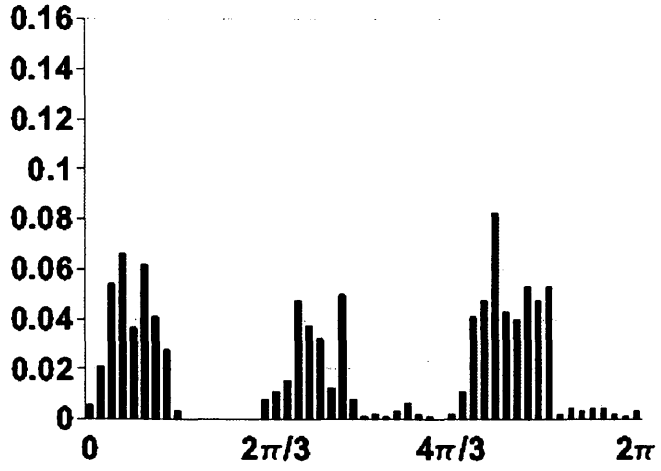


Figure 2.4: Histogram of phase values computed with a sliding window on the sequence from Figure 2.1 with two bases deleted creating three reading frames.

The Fourier magnitude value is:

$$\frac{1}{2}\left[(C_0 - C_1)^2 + (C_1 - C_2)^2 + (C_2 - C_0)^2\right]. \tag{2.5}$$

A disadvantage of using the DFT to analyze DNA sequences is that, in order for it to work well, it is necessary to use a window with length of at least a few hundred base pairs. This means that it is not useful for characterizing short sequences.

2.1.3 Autoregressive Models

A technique that works well for short sequences is an autoregressive model [55]. Autoregressive models are used for analysis of genomic sequences in [33]. They have the advantage over the DFT in that they work with smaller window sizes and, thus, shorter sequences. The idea of a forward predictive autoregressive model is that, given a number p of previous values in a sequence x, the value of $x(n)$ can be predicted using:

$$x(n) = \sum_{k=1}^{p} a_k x(n-k) - e(n) \tag{2.6}$$

where $a_1, a_2, \ldots, a_p$ are prediction coefficients, $e(n)$ is the prediction error, and p is the order of the model. Likewise, a backward predictive AR model predicts the value of $x(n)$ based on following values in the sequence. The prediction coefficients are calculated by minimizing the mean squared forward prediction error by solving the Yule-Walker equations or by using the Burg Method.

The Yule-Walker equations are:

$$\sum_{k=1}^{p} a_k r_{xx}(i-k) = r_{xx}(i), i = 1, 2, \ldots, p \tag{2.7}$$

where r_{xx} is the autocorrelation function. Since the sequences are not infinite, an estimator must be used for evaluating the autocorrelation. Reference [33] uses:

$$\hat{r}_{xx}(i) = \frac{1}{N} \sum_{k=0}^{N-|i|-1} x(k+|i|)x(k) \tag{2.8}$$

where N is the length of the window.

The Burg Method is based on the Levinson-Durbin recursion algorithm for solving the Yule-Walker equations. In order to get a more stable solution, Burg's Method minimizes not just the forward prediction error, but the sum of the forward and backward prediction errors.

Once the prediction coefficients have been calculated, they can be used in various ways. One way is for comparing sequences. A model for one sequence (for example, a gene) can be computed and then used to calculate the error for another sequence (one being tested to see if it is a gene) using that model. This gives a measure of "goodness of fit." In [33] it was found that the "goodness of fit" test did not work well for gene prediction and that it was highly specific for particular genes, especially as the model order increased. An alternative is to use the prediction coefficients as features of the sequence. This is more useful. In [33] it produced good results for gene finding, looking for repeated sequences, and identifying sequences with similar chemical structures.

Autoregressive models were also used for finding tandem repeats (short sequences repeated many times in a row) in [154] and for classifying HIV-1 subtypes in [150]. In [154], peaks in the power spectral density function $P(\omega)$ calculated from the prediction coefficients for the model using:

$$P(\omega) = \frac{\sigma^2}{|1 + \sum_{k=1}^{p} a_k \exp(-j\omega k)|^2} \tag{2.9}$$

indicated period m repeats, where the peak value $\omega = \frac{2\pi}{m}$. In [150] an artificial neural network trained on prediction coefficients was used for classification.

2.1.4 String Kernel

Instead of converting DNA sequences to numerical values, some researchers study them using kernel methods, such as Support Vector Machines (SVMs). A kernel provides a method for mapping data from one space to another in order to perform the "kernel trick" of separating data that is unseparable in the original space. For more information about SVMs, see Section 6.27.

To perform the kernel trick, it is necessary to define a kernel that operates on strings. One way to do this is by using k-mers. A k-mer is a string of length k generated from an alphabet. For DNA sequences either the alphabet $\{A, C, G, T\}$ or the alphabet containing the 20 amino acids can be used. Typically, sequences are counted from all possible starting points in the sequence, so that, for example, the sequence AGGT contains the 2-mers: AG, GG, and, GT. To form a string kernel, one calculates the frequency of

occurrence of all k-mers for a given k, for example all 3-mers. Sometimes all k-mers for $k = 1..n$ for some particular n are used. Some examples of successful applications of k-mers to bioinformatics classification problems include [76, 69, 5, 2]. String kernels (using different alphabets) are also used for classification problems, for example text classification.

String kernel features have the advantage that they require no biological knowledge to construct, and they yield a large set of features, some of which are often effective. They have the disadvantage that, as k increases, the number of features increases exponentially. Also, for large k, many of the features have a value of zero. For example, if k is six, the expected value of the 6-mer feature, AACGGT, in a random sequence of length 200 is 0.05. Also, insertions and deletions are common in DNA sequences. The string "TTTTTTTT" often has the same biological significance as the string "TTTTTTTTTTTTTTT," but the 4-mer "TTTT" occurs five times in one and twelve times in the other.

The string kernel features supplement the other statistical features in the classifiers. They are also used as a basis for comparison of for the SEM features (Chapter 4), since SEM features also have the property of creating a large set of potentially useful features. They can be used with or without feature selection.

2.1.5 Entropy

A measure from information theory that has been useful in gene finding is Shannon entropy [119]. It is useful because of the fact that in protein coding regions not all codons (groups of three nucleotides coding for an amino acid) are used uniformly, while in non-protein coding regions they are. In [22] Bernaola-Galvan et al. compute the entropy of a sequence using a 12-symbol alphabet $\{A_0, A_1, A_2, C_0, C_1, C_2, G_0, G_1, G_2, T_0, T_1, T_2\}$. The letters A, C, G, and T represent the four possible bases; the subscripts represent their position in the sequence mod 3, i.e. their position in their codon. Other alphabets are possible: the 4-symbol alphabet of bases, the 16-symbol alphabet of dinucleotides (pairs of bases occurring in a row), the 64-symbol alphabet of trinucleotides. For each sequence in [22], the frequency vector $F = \{f_1, \ldots, f_{12}\}$ was computed for the 12 symbols. Shannon entropy $H(F)$ was calculated using the formula:

$$H(F) = -\sum_j f_j \log_2 f_j \tag{2.10}$$

To compare two sequences, the Jensen-Shannon divergence $C(F_1, F_2)$ was calculated using the frequency vectors (F_1 for the first sequence, F_2 for the second sequence, and F for the concatenated sequence), lengths (n_1 for the first sequence, n_2 for the second sequence and N for the concatenated sequence), and the formula:

$$C(F_1, F_2) = 2 \ln 2[NH(F) - n_1 H(F_1) - n_2 H(F_2)]. \tag{2.11}$$

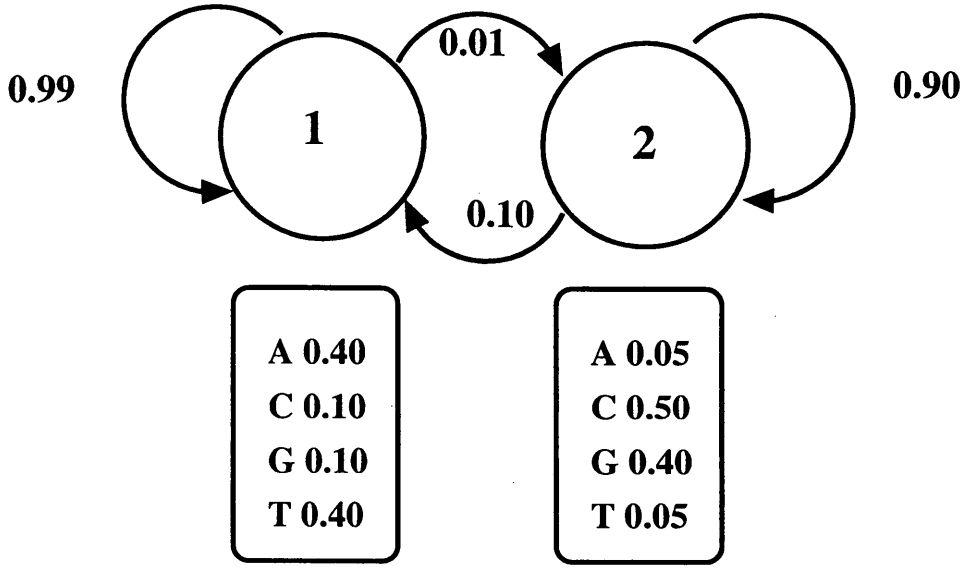
Two sequences are considered to be of different types if their Jensen-Shannon divergence is greater than that of two random sequences.

This method was used to find the boundaries between protein coding and non-protein coding regions in the following way. A sliding pointer was moved through the genome and the Jensen-Shannon divergence was calculated for the sequences on either side of it. The point of maximum divergence was calculated. If that divergence was greater than for random sequences, a cut was made. Next, the procedure was repeated on the subsequences created. There resulted in a segmentation of the genome that was a good match for coding/non-coding boundaries.

Another way to use Shannon entropy is to calculate based on the string kernel. It can be calculated, for example, for dinucleotides, trinucleotides, or both grouped together. In this way, it acts as a sort of summary feature for the string kernel features, thus reducing the number of features while retaining much of the information. For example, the single 6-mer entropy feature summarizes the 4096 6-mer features in the string kernel.

2.1.6 Hidden Markov Model

Another way of modelling a DNA sequence type is with a Hidden Markov Model (HMM) [63]. HMM are used in many other applications as well, such as musical score following and handwriting recognition. A HMM looks like a probabilistic finite state machine. There is a state for each base in the sequence and four transitions out of that state, one



Sequence (input)

A	A	T	G	T	C	C	C	G	A	T	...
0.40	0.40	0.40	0.10	0.40	0.50	0.50	0.50	0.40	0.40	0.40	

States (hidden)

1	1	1	1	1	2	2	2	2	1	1	...
1.00	0.99	0.99	0.99	0.99	0.01	0.90	0.90	0.90	0.10	0.99	

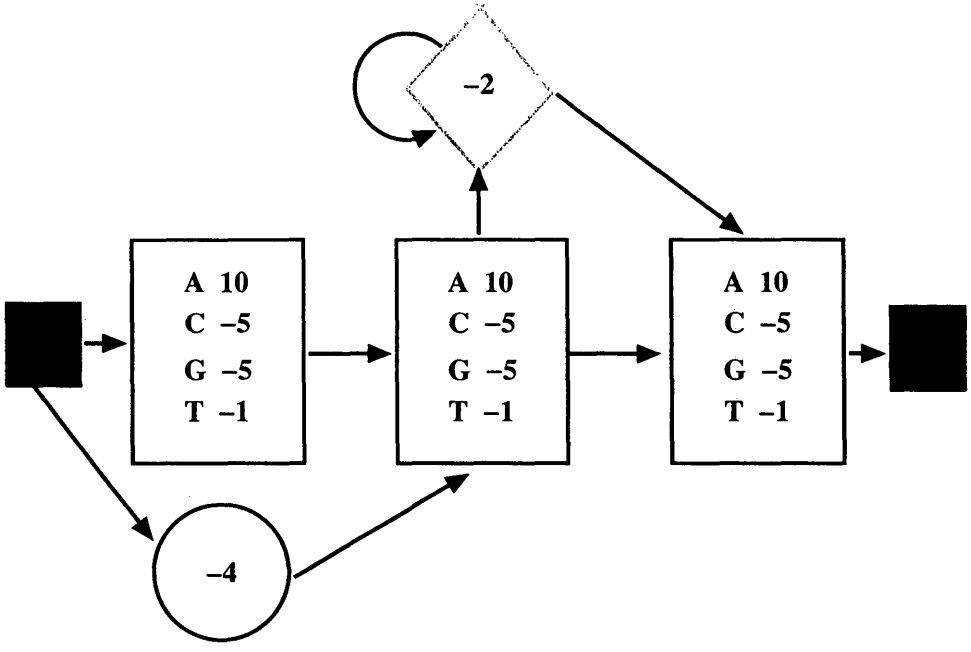
Figure 2.5: Example of a Hidden Markov Model.

for each possible base, labelled with the probability that base will occur. An example is shown in Figure 2.5. The probabilities of the possible paths are used to calculate a score for a sequence based on the probability the model has of generating that sequence. The word "hidden" refers to the sequence of states followed to generate the sequence. In contrast to simpler Markov models, like Markov chains, these are unknown and have to be determined. An expectation-maximization algorithm is used to find the sequence of hidden states for a given input. It has complexity $O(L * S^2)$ where L is the length of the sequence and S is the number of states. Thus, for detecting long, complicated sequences its use is impractical. It has, however, been used successfully for gene finding [56].

A variation on a HMM is a profile Hidden Markov Model (pHMM) [63]. An example is shown in Figure 2.6. This model is used to identify a particular sequence that has been modified by mutation with insertions, deletions, and substitutions. It has three types of states: match states, insert states, and delete states. A score is calculated for a sequence based on the best path through the pHMM. For a review of the use of various types of HMM in bioinformatics, see [151]. An example of its use detecting a type of TEs is described in Section 2.2.3.

2.2 Related Work Detecting And Classifying TEs

Current annotations of TEs are based on a variety of bioinformatic tools and are considered to be works in progress. RepBase contains a library of known TEs. A general



Sequence: G A G G A

Score: −4 +10 − 2 − 2 +10 = 12

Figure 2.6: Example of a profile Hidden Markov Model. Green squares represent the start and end; green rectangles are states representing each position in the sequence; the yellow diamond is an insert state; the red circle is a delete state.

purpose tool for identifying TEs is RepeatMasker [121]. It works for all types of TEs and a variety of species, any that have TEs in the RepBase library. Tools specifically developed for finding HERVs include HERVd [98, 99], RetroSearch [139, 1, 70], and RetroTector [124, 123]. There are also a variety of tools designed to detect TEs in non-human species. These are of value to this work both for creating data sets for machine learning and for providing ground truth for testing. Fewer previous works have focused on distinguishing different types of TEs. The work in this thesis supplements that done by TEclass [2] and REPCLASS [47].

2.2.1 RepeatMasker

RepeatMasker [121] is a program designed to screen DNA sequences for repeated elements (including ERVs) and for low complexity sequences (like a 100 base pair sequence that is mostly As and Ts). About half the human genome falls into these categories. Users of RepeatMasker have to identify the species of the input sequence. It operates based on sequence homology with reference consensus sequences. Best results are obtained for human and mouse sequences since these currently have the best collections of reference consensus sequences. RepeatMasker's primary purpose is to mask parts of a DNA sequence whose presence could result in false positive matches in another search. Since it categorizes the type of repeats it finds, it is also useful for those interested in a particular type of repeat, like all ERVs or all occurrances of a specific type of ERV.

RepeatMasker starts with a database of consensus sequences for each repeat type taken from RepBase. A dynamic programming algorithm, called the Smith-Waterman-Gotoh algorithm [122, 51] is used to align the consensus sequences to the input sequence. Sequence alignment is challenging, because mutations can cause substitutions, insertions, and deletions in sequences. In order to get the best match, it is necessary to decide where to put gaps and how big to make them. Also, if the sequences are different lengths, there can be many choices of where to start the alignment. The Smith-Waterman-Gotoh algorithm finds the optimal alignment by constructing a matrix with scores for possible alignments. Scores for alignments are based on a weighting system designed so that matches improve the score, mismatches detract from it, and there is a penalty for gaps. The score at position (i, j) in the matrix represents the best possible score for the first i bases in sequence 1 and the first j bases in sequence 2. The optimal alignment can be constructed by backtracking from the highest score in the matrix.

2.2.2 HERVd

HERVd[3] was a database of HERVs. It was last updated in 2003, was operational when this research was begun, but is no longer being maintained. It is based on the build of the human genome current in 2003. The user can search the database for a specific HERV (using its ID number), for HERVs with a specified range of lengths, for HERVs

[3] `http://herv.img.cas.cz/`

in a particular family, for HERVs on a particular chromosome or in a particular section of a chromosome, for HERVs with a given orientation (sense or antisense), or for HERVs with a specified GC content (proportion of G and C bases), or for any combination of these. The original database included 39 HERV families and identified about a third of HERVs. The number of HERVs identified was increased when more consensus sequences were identified, doubling the number of HERV families included.

The database was assembled starting with the consensus sequences for various families of HERVs in RepBase. RepeatMasker was used to screen for non-retroviral transposons that would confuse the search and to search for matches to the consensus sequences. To cope with identification problems caused by mutations resulting in insertions and deletions (including retroviruses inserted into other retroviruses), HERVd employs a defragmentation algorithm. The creators of HERVd do not provide complete details of how this algorithm is implemented. All that is said is that it uses profile Hidden Markov Models [44] to define HERV families and to assign HERV sequences to these families. The defragmentation algorithm pieces together the sequence fragments identified by RepeatMasker. In addition, the flanking DNA is examined in an attempt to identify target site duplications (TSDs). TSDs are copies of a small number of bases at the insertion site of a transposon.

Advantages of this method include having the identified retroviruses classified by family and being able to find retroviruses that are heavily mutated and fragmented. The

major disadvantage is that the process must be repeated every time a new build of the genome is done and that it must be custom designed for a particular species (in this case, human beings). Another disadvantage is that it is only able to find retroviruses that belong to known families.

The HERVd database has been used in various ways by researchers. Some of the authors of the database, together with some other researchers, used it to study the HERV-W family [100]. The HERV-W family is important, because of its possible role in multiple sclerosis, rheumatoid arthritis, and schizophrenia. Their studies suggested that scientists studying these diseases should focus on a particular subset of HERV-Ws. In [29], the HERVd database was used to study the sequence CCTGTT, a sequence that occurs unusually often in the human genome. The authors used HERVd to discover that this sequence occurs even more often in HERVs. Belshaw et. al. [19] used HERVd to study mutations in the *env* gene of various HERV families. Since the *env* gene is only needed if the retrovirus leaves the cell, the types of mutations found in it are a clue to whether copies of the retrovirus were created by infection from exogenous viruses or from retrotransposition (copying and pasting the new copies in the genome).

2.2.3 RetroSearch

Another database of HERVs is called RetroSearch[4]. RetroSearch lets users search for HERVs in the 2003 Human Genome Assembly by ID number, HERV family, location (a range can be specified), minimum length, minimum ORF length, minimum number of ORFs, minimum identity to a known retroviral protein, minimum LTR length, a specified range of distances between LTRs, and minimum identity of LTRs to known LTRs. ORFs (open reading frames) are regions coding for particular proteins. When referring to retroviruses, the terms "gene" and "ORF" are often used interchangeably (although many biologists consider this incorrect).

The RetroSearch database was built starting with a query database of 237 endogenous and exogenous retroviruses from a variety of host organisms. These sequences were edited so that they contained only the part of the retrovirus that codes for genes. This was to avoid finding solitary LTRs. A BLAST search of the human genome from the query database was done. The BLAST algorithm is similar to the Smith-Waterman-Gotoh algorithm used by RepeatMasker, except that it uses a heuristic instead of an exhaustive search. This allows it to run about 50 times faster at the cost of some accuracy. It identifies matches (hits) between the query and the input sequence and assigns scores. Overlapping hits were clustered together and assigned a region score based on the BLAST scores of the sequences. If the score was high enough, the flanking DNA

[4] www.retrosearch.dk

on either side was examined for LTRs. The results were called "putative HERVs". The next step was to find ORFs in the putative HERVs. This was done by searching for stop codons within the putative HERVs. Regions between two stop codons that were long enough ($>$ 62 nucleotides) were compared to a database of over 6000 retroviral and non-retroviral proteins using a FASTA search [81, 101]. A FASTA search is similar to a BLAST search, except that it is especially tuned for aligning proteins. Regions with retroviral ORFs were identified as HERVs. The original database using this method identified 1.1% of the human genome as containing HERVs (about 14% of HERVs). This database was later updated and expanded. Data is displayed online together with data for the same regions from RepeatMasker. Often, there are noticeable differences. In the course of this research, RetroSearch was removed from the internet due to becoming outdated.

Advantages of this method include having the HERV ORFs identified and searchable. One can, for example, search for all HERVs that have an *env* ORF that is more than 200 nucleotides long. (It finds 493 of these.) This makes it possible to assemble custom databases for studying particular retroviral genes. This database is pickier about its choices than HERVd, identifying fewer, but doing more to verify that the retroviruses in it are actually retroviruses and not just sequences that resemble retroviruses. The disadvantages are similar to those of HERVd: the process must be repeated for every new build of the genome sequence, it only includes human ERVs, and only retroviruses

similar to known retroviruses are found.

RetroSearch has been used primarily for studying intact retroviral genes. In particular, it was used to study a HERV envelope gene that is expressed in the placentas of monkeys and apes [70].

2.2.4 RetroTector

The RetroTector algorithm[5] takes as input a genomic sequence of length 5,000 to 10,000,000 bps. It was originally designed only for human genomic sequences, but has been extended to accommodate primate, chicken, cow, dog, elephant, horse, lizard, mouse, and zebra fish genomes. However, the species must be given as input to the algorithm. Retrotector scans the input sequence, finds ERVs, identifies the LTRs and retroviral protein genes, and outputs them in an annotated format such as is found in textbooks.

The algorithm has five parts:

1. First, it "sweeps" the input sequence, masking out all Alu and L1 fragments, since these could be confused with ERVs. Alu is the most common SINE, and L1 is the most common LINE.

2. Next, it searches for LTR pairs and attempts to find solitary LTRs. Solitary LTRs are usually remnants of a cut-and-paste transposition in which the "cut" is incomplete.

[5] `http://www.kvir.uu.se/RetroTector/RetroTectorProject.html`

3. Then, it searches for motifs (using a library of 275). Motif is defined loosely. From a programmer's point of view, a motif is a procedure for detecting a conserved ERV trait, taking into account the possibility of mutation. Most of the motifs are procedures that detect close matches to specified amino acid sequences, but they also use trained neural nets and some other procedures. The program is designed so that it is easy to add new motif modules. Each motif has constraints on its relationship to other motif hits and the LTRs (distance from and relative position). Also, each motif is assigned to a particular retrovirus genera. Using this information, the algorithm puts together chains of motifs and LTRs, creating a putative ERV. This is the part of the algorithm that is species specific. New motifs and constraints have to be devised for each species.

4. Given the putative ERV, the algorithm tries to identify the genes for *gag, pro, pol,* and *env* proteins. These are proteins common to all retroviruses.

5. The last step is to look for other possible genes.

Retrotector has been used to build a retroviral taxonomy [61], to study HERV gene expression in cancer [3], and to study the impact of retroviral defence mechanisms, known to disable exogenous retroviruses, against endogenous retroviruses [62].

Researchers associated with the Retrotector project use profile Hidden Markov Mod-

els (pHMMs) to detect solitary LTRs in [20]. They use different models for HML[6], gamma, beta, and lenti retroviruses, as well as a general model. Their best model, which detects HML retrovirus solitary LTRs, achieves 87% sensitivity and 96% specificity (92% accuracy), and their combined models achieve 53% sensitivity and 74% specificity (64% accuracy) on a scan of human chromosome 19 as compared to RepeatMasker annotations. Their actual accuracies could be much higher as the RepeatMasker annotations are unlikely to be entirely correct and complete. From their models they identify seven conserved modules in solitary LTRs.

2.2.5 ERVs In Non-human Species

Researchers focusing on organisms other than human beings prefer different terminology. Instead of "endogenous retroviruses" they say "LTR retrotransposons" (a somewhat broader category), and the focus is more often directed towards discovering how they affect the operation and evolution of the genome rather than on how they impact disease. The researchers are interested not in specifically finding ERVs, but in finding all repeated elements, all TEs, or all retrotransposons (SINEs, LINEs, and ERVs). Another distinction they make is between Class I and Class II transposable elements. Class 1 TEs do not use an RNA intermediate when they replicate. Class II TEs do use RNA intermediates and are also called "retrotransposons" or "retroelements". Other common

[6]HML retroviruses are beta retroviruses that have been well-studied in humans because they include the most recent insertions. They are families of HERV-K. HML stands for Human MMTV-like.

distinctions are "replicative" (copy-and-paste transposition) and "nonreplicative" (cut-and-paste transposition) and "autonomous" (encode genes for replication) and "nonautonomous" (use genes from other TEs to transpose). See a good genetics textbook, such as [77] or [104], for more information.

There are many software tools designed to perform these tasks. For a partial list, see Table 2.1. For a review, see [21]. There are four different approaches: repeat finding methods, homology-based methods, structure-based methods, and comparative genomic methods. Table 2.1 lists some of the tools along with which methods they use. Repeat finding methods look for repeated sequences in the genome. They use computational strategies such as suffix trees and hashing. Homology-based methods take advantage of prior knowledge by comparing sequence fragments to a database of classified sequences. Some use a direct comparison with the sequences in the database; others compare the sequences using profile Hidden Markov Models. Structure-based methods use known characteristics of the structure of the elements to find them. They look for characteristic features of pieces of the ERV: LTRs, TSDs, PBSs, PPTs, and sequences that code for proteins found in the elements. LTRs are repeated regions at the beginning and end of the ERV. TSDs are short (4-6 bp) repeats in the region flanking the ERV on either end. PBSs are primer binding sites, the place on the ERV where transcription starts. PPTs are polypurine tracts, a section of the ERV rich in purines, A and G bases . (ERV structure is discussed in more detail in Section 3.1.1.) They identify subsequences that have these

features spaced appropriately. Comparative genomic methods compare closely related genomes, either from the same species or closely related species. Regions that exist in one genome but not in others are likely transposons. After a new transposon family has been discovered, a consensus sequence is created and put in RepBase to be used in future searches using homology-based methods.

It seems odd that so many tools have been created, especially since they are freely shared. Why not just develop one good tool and use it? The reason is that each tool has strengths and weaknesses, and the best results are obtained by using a combination of them. Saha et. al. [115] compared six *de novo* repeat finding algorithms using the same data from the rice genome and found that their results were profoundly different. *De novo* algorithms are algorithms that do not incorporate a database of known elements. In [64, 57, 107] there are examples of studies in which a combination of tools were used effectively in the rice, chicken, and fruit fly genomes.

Quesneville et. al. [107] describe the process by which the transposons in the Drosophila genome were annotated. Since *Drosophila* was the first large genome sequenced, it is of particularly high quality and its assembly has been well verified, making it a good choice for developing a model process. A pipeline for annotating the transposons in Release 4 of the Drosophila genome was developed using the manually curated set of annotations from the Release 3 genome as a benchmark to test the technique. The results using four homology methods and four *de novo* methods were compiled and given

Table 2.1: Software tools for TE discovery.

name	method	website
SSAHA [94]	repeat finding	www.sanger.ac.uk/resources/software/ssaha/
REPuter [75, 74]	repeat finding	bibiserv.techfak.uni-bielefeld.de/reputer/
ReAS [79]	repeat finding	ftp.genomics.org.cn/pub/ReAS/software/
RepeatScout [106]	repeat finding	bix.ucsd.edu/repeatscout/
BLAST [6]	homology	blast.ncbi.nlm.nih.gov/Blast.cgi
HMMER [43]	homology	hmmer.janelia.org/
MGEscan-LTR [111]	structure	darwin.informatics.indiana.edu/cgi-bin/evolution/ltr.pl
LTR_STRUC [91]	structure	www.mcdonaldlab.biology.gatech.edu/finalLTR.htm
LTR_par [66]	structure	www.eecs.wsu.edu/~ananth/software.htm
LTR_FINDER [148]	structure	tlife.fudan.edu.cn/ltr_finder/
LTRdigest [125]	structure	genometools.org
Capsi and Pachter [32]	comparative genomic	math.berkeley.edu/~lpachter/software.html

to five human curators, each working on a separate chromosome arm. A single human curator did a second pass to improve consistency. The fact that to get a high quality annotation eight methods were needed plus manual curators demonstrates the importance of developing new methods using novel approaches.

2.2.6 Related Work Classifying Different Types Of TEs

Although there are many software packages designed to detect genomic repeats, there are few designed to classify already detected repeats. Two such are TEclass [2] and REPCLASS [47]. TEclass classifies based on k-mer statistics. Elements are classified as DNA transposons, LTR retrotransposons, SINEs, or LINEs. Three classification methods are used: SVMs, random forests, and learning vector quantization.

REPCLASS classifies LTR retrotransposons, DNA transposons, SINEs, LINEs, and Helitron elements[7] using three modules: a Homology Module, a Structural Module, and a TSD (target site duplication) Module. Their Homology Module compares the sequences to the RepBase repeat library. Their Structural Module looks for structural features typical of particular types of repeats. REPCLASS's TSD Module examines the repeated elements in their genomic context, looking for target site duplications, which are short sequences repeated at the beginning and end of some types of repeats that are different for each instance of the repeat. The creators of REPCLASS do not test their method on solitary LTRs and say in [47] that they suspect it will not work well for them.

[7]Helitron elements are DNA transposons that transpose by rolling-circle replication instead of the usual cut-and-paste replication.

2.2.7 Difficulties Created By Sequence Assembly Methods

The retroviral detection methods described in the previous sections operate on assembled sequences. A limitation of their effectiveness results from the way sequence assembly is done. With current technology, the longest DNA strand whose sequence can be directly determined is 1000 bps long. The shortest human chromosome is 50,000,000 bps. In order to sequence a chromosome, a process called shotgun sequencing is done. In shotgun sequencing, the DNA is randomly shattered into pieces using ultrasound, the pieces are inserted into cloning vectors (known sequences of DNA), the cloning vectors are inserted into a bacteria, multiplied until there are enough to sequence, sequenced, and then the pieces are put them together like a jigsaw puzzle. The putting together step is called sequence assembly. Typically, the genome is oversampled by 20-30 times so that little is missed. Sequence assembly is not a trivial task and the primary reason is the repeated regions, like SINEs, LINEs, and ERVs. It is like putting together a jigsaw puzzle in which large numbers of the pieces are identical. The assembled sequence also includes gaps, due to the fact that the sampling is not really random; the cloning vectors prefer some pieces over others. The assembly process is described in [93, 105, 116]. The consequences of using assembled sequences for detecting TEs are that the TEs are likely to be included in the unsequenced gaps (for example, if they are toxic to the bacteria in which they are grown), are likely to be put in the wrong place, and some copies are likely

to be left out entirely.

2.3 Conclusion

This chapter reviewed statistical and signal processing methods for analyzing DNA sequences: DFT, autoregressive models, string kernels, and entropy. The next chapter will present some novel statistical features based on the DFT designed specifically for detecting TEs. The string kernel and entropy features in Chapters 6 and 7 will be used together with the features presented in Chapters 3 and 4 to build classifiers to identify TEs and distinguish TEs from other sequences. Autoregressive models will not be used because they proved to be ineffective with TEs due to their non-linear character.

This chapter also reviews a machine learning method for DNA sequence analysis: pHMMs. String kernels are also used with machine learning. Chapters 4 and 5 will apply anther machine learning method, SEMs (Section 1.1.2), to the problem of DNA sequence analysis. The string kernel will be used as groundwork for understanding and analyzing the features produced by SEMs. The databases discussed in Section 2.2 will be used to create training sets and also to quantify the success of the methods.

3 Statistical Features

Three novel sets of statistical features for the detection and classification of TE sequences are presented. The first set of features uses the fact that overlapping genes are often found in LTR retrotransposons and ERVs and these genes typically use more than one reading frame. The reading frame predictions made using the Fourier transform are used fto generate features that capture the reading frame structure of LTR retrotransposons and ERVs. The second set of features characterizes the non-randomness of the sequences based on their frameshifts. The final set was designed to find patterns in TEs that are regulatory rather than protein coding and to aid in interpretation of SEM features. This feature set is based on statistics of the bases in the sequences.

3.1 Reading Frame Structure Features

The first set of features were developed to identify TEs that are remnants of retroviral insertions, such as LTR retrotransposons, ERVs, and IESs. These features are based on the observation that retroviral DNA often has overlapping genes that use multiple reading frames. It is difficult to predict where and how these frameshifts occur. Even though all

retroviruses have the same three (*gag*, *pol*, *env*) genes always occurring in this order, the sequences of these genes vary so much that it is necessary to use a large library in order to identify them with sequence homology techniques. Even with a large library, many sequences are missed. Therefore, an indirect approach was used: estimate the frameshifts and then use machine learning to determine whether that pattern of frameshifts suggests the sequence is a retroviral-descended TE. Fourier analysis was used in a way similar to that done in [90], described in Section 2.1.2.

3.1.1 Retroviral Genomic Structure

Retroviral DNA is incredibly diverse. Retrovirusus have a mutation rate much higher than that of other DNA. Also, many ERVs that integrated into our genome millions of years ago are now in inactive portions of the genome. Hence, there is no selection pressure governing their mutations. This means that the task of finding the ERVs in a DNA strand is not trivial. Methods currently used to find them include looking for repeated elements [98], comparing to known retroviral genomes [139], and using biological knowledge combined with machine learning [124]. None of these methods is perfect. Algorithms that search for repeated elements may miss ERVs that are not repeated often; algorithms that compare to known retroviral genomes may miss ERVs if the database is incomplete; and algorithms that use biological knowledge may miss ERVs with unusual characteristics and only work well if they are used on the genomes of organisms

for which they have been trained. Each method is hampered by the fact that many ERVs are heavily mutated, but the mutations affect each algorithm in a different way.

One difference between retroviral genomes and genes of eukaryotes (plants and animals) is that eukaryotic genes typically use only one open reading frame (ORF), while retroviruses typically use all three ORFs on a DNA strand. These ORFs sometimes overlap. Furthermore, all retroviral genomes have the same basic pattern with minor variations for different types. They begin and end with non-coding sequences, and they have three genes that always occur in the same order (*gag* , *pol*, and *env*). Some retroviruses have additional genes; in particular, lentiviruses (like HIV) have six additional genes.

Retroviruses are difficult to detect using sequence homology, i.e., by looking for sequences similar to known retroviral sequences, because they are diverse, having only small portions of their genomes in common. For example, it is estimated that there are 10^{60} variants of HIV [138]. An analysis of HIV-1 copies within a single nucleus showed 28% variation in the *env* gene [142], more than the average protein variation between birds and humans. One explanation for this variability is that when RNA is converted to DNA using reverse transcriptase, there is no error correction such as there is when DNA is copied. Also, due to a lack of selective pressure, many ERVs are heavily mutated to the point of being defective or even completely non-functional. These defective ERVs are still of interest, however, because of their past influence on the genome, their value as molecular fossils, and because some of them can function with the help of other ERVs.

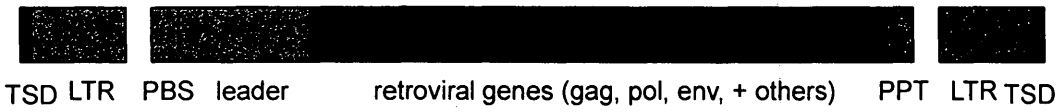
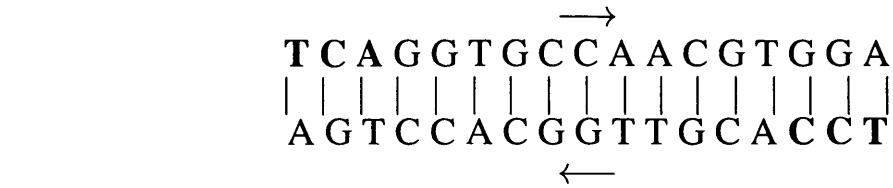


Figure 3.1: Retrovirus Structure

An alternative to detection using sequence homology is detection based on structure. See Figure 3.1 for an illustration of the retroviral structure. They range in size from 5000 to 20,000 nucleotides. An intact retrovirus begins and ends with a 18-250 bp LTR. Another short non-coding sequence follows. There is then an 18 nucleotide primer binding site (PBS). A somewhat longer (90-500 nucleotides) sequence, called a leader, follows. Then, come the genes. After the genes, there is a very short (about 10 nucleotides) sequence called the polypurine tract (PPT). Then there is a short non-coding sequence, followed by the LTR. In addition, the DNA flanking the ERV often includes a TSD consisting of the 4-6 bases at the insertion point. Copies of these bases can be found on either side of the inserted retrovirus.

Retroviral genes have an unusual structural feature – they use overlapping reading frames. Reading frames in DNA arise from the fact that the genetic code maps nucleotides onto proteins in groups of three. This means that the code translates differently depending on whether decoding begins at position 0, position 1, or position 2. For an illustration of this see Figure 3.2. Starting at position 3 will have the same result as



Reading Frame 1: TCA GGT GCC AAC GTG GA?→ SGANV ...
Reading Frame 2: CAG GTG CCA ACG TGG A??→ QVPTW ...
Reading Frame 3: AGG TGC CAA CGT GGA ???→ RCQRG ...
Reading Frame 4: TCC ACG TTG GCA CCT GA?→STLAP ...
Reading Frame 5: CCA CGT TGG CAC CTG A??→PRWHL ...
Reading Frame 6: CAC GTT GGC ACC TGA ???→HVGT{STOP} ...

Figure 3.2: Decoding using different reading frames. The DNA strand is broken into codons on the left, and the symbols on the right represent amino acids.

starting at position 0 (excluding the first protein), so the codes starting at position 0 and position 3 are said to be in the same reading frame. On any segment of DNA, there are six possible reading frames, three in each direction (sense and antisense). The genes in the cells that host the retroviruses mostly use a single reading frame. Retroviral genes, however, use all three reading frames in a given direction with the end of one gene often overlapping the beginning of another.

3.1.2 Fourier analysis

To encode the DNA sequences so they can be used with the Fourier transform, binary indicator sequences inspired by those used in Z-curves (see Section 2.1.1) were used. Three binary strings are created: the RY string has a value of one for R bases (purines) and zero

for Y bases (pyrimidines); the MK string has a value of one for M bases (aminos) and zero for K bases (ketos); the SW string has a value of one for S bases (strong H-bond) and zero for W bases (weak H-bond).

The DFT produces a complex number which can be divided into a magnitude and a phase. For these feature, the phase values are used. In [90] it was shown that an insertion or deletion in coding regions shifts the reading frame by $-\frac{2\pi}{3}$ and $\frac{2\pi}{3}$ respectively. When a sequence uses more than one reading frame, the Fourier phase histogram shows three peaks spaced $\frac{2\pi}{3}$ apart similar to those in Figure 2.4. Fourier phase values at relative frequency 1/3 are quick and easy to calculate using Equation 2.4, making features based on them computationally cheap.

Once the Fourier phase histogram for a large region (i.e., the size of a typical ERV) has been calculated, the reading frame for any subsequence of that sequence can be estimated based on where its Fourier phase falls in the histogram. So, for example, if the Fourier phase histogram showed reading frame 1 to be between $\frac{\pi}{3}$ and π, then a subsequence with Fourier phase $\frac{\pi}{2}$ would be estimated to be in reading frame 1. This estimate would be meaningless if the subsequence was non-coding, and it might be wrong even for a coding subsequence because the subsequence had unusual statistical properties. This could happen, for example, in regions that code for overlapping reading frames. This technique, thus, as well as estimating the reading frame, detects regions with unusual statistical properties relating to the reading frame.

3.1.3 Fourier phase vectors

Fourier phase vectors are strings created using combined reading frame information from some or all of the indicator sequences. For these features just the RY and SW strings are used. This is because, for this purpose, the MK string does not seem to contribute useful information. The reading frame with the most members is designated Reading Frame 0; Reading Frame 1 is shifted one nucleotide from Reading Frame 0; Reading Frame 2 is shifted two nucleotides. A sliding window estimates the reading frame value for each string and combines them base 3. For example, a 3 (10 base 3) in the string means that the RY string reading frame estimate for that window is 0, and the SW estimate is 1. Mixed signals are common in non-coding or overlapping regions. The strings of reading frame integers created as described above are of lengths equal to the number of positions of the sliding window used to compute them.

A small amount of crucial information is gathered from these Fourier phase vectors in the following way. Identical successive values indicate homogeneity in the sequence, at least in terms of reading frames. Disagreements signify putative changes, though not all putative changes are frameshifts. The relative frequency of these changes over sequences of a given length forms a key feature of the sequence. This property is encapsulated in terms of a short vector as follows. Given a sequence of phase vector values (encoded as integers), the positions in the sequence where successive values differ are computed –

these are called *change points*. Then the distances between every two successive change points are computed. Let $f(i)$ be the frequency of distance i observed in a sequence. Four values are used as features. These values correspond to the expressions $f(1)$, $f(2)$, $f(3)+f(4)+f(5)$ and $f(6)+f(7)+f(8)+f(9)$. Non-coding sequences tend to have large values for small i, meaning the sequences switch reading frames often; coding sequences have smaller values, meaning the sequence usually stays in the same reading frame for stretches longer than i bp. In addition, sequences with overlapping coding regions (like ERVs) have short segments representing those overlaps. The average distance between change points is also computed and used as a feature. This value tends to be larger in sequences with coding regions, and is similar for sequences that have coding regions of similar size and placement.

3.2 Frameshift Histogram Features

The second set of features detects randomness in the sequence based on the Fourier phase histogram. The motivation for its use was to distinguish TEs with coding regions from non-coding intergenic sequence. Retroviruses (and thus retrovirus-descended TEs) typically use all three reading frames in a given direction; genes typically use just one reading frame; non-coding sequences have their own kind of distinctive Fourier phase histogram. An example of a phase histogram for a coding region is shown in Figure 2.1, for a non-coding region in Figure 2.2 and for a retrovirus in Figure 3.3. Notice that

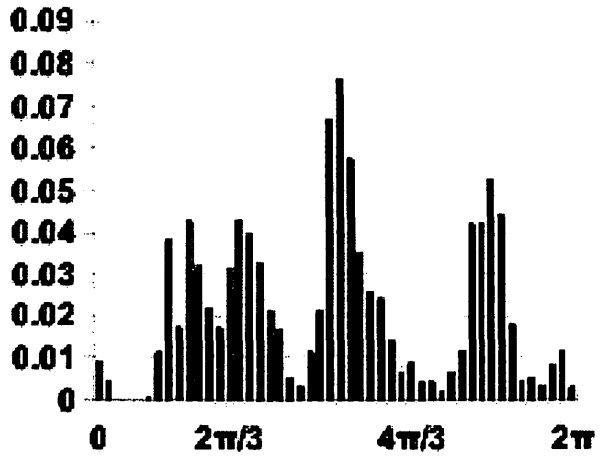


Figure 3.3: Histogram of phase values computed with a sliding window on the T sequence of the complete genome of the enzootic nasal tumour virus of goats (reprinted from [16]).

the retrovirus histogram in Figure 3.3 has three identifiable regions of width $2\pi/3$ that look similar to the histogram in Figure 2.1. Each region contains phase values for parts of the sequence that code for the same reading frame. The pattern is not always this clear. Figure 3.4 shows the phase histogram for another retrovirus for which the pattern is not so clear. The pattern is often even less clear, but still somewhat apparent, in LTR retrotransposons and ERVs due to mutation.

3.2.1 Frameshift Histograms for Random Sequences

The Fourier phase histograms for random sequences are interesting and deserve some further discussion. Intergenic non-coding sequences sometimes consist of regulatory sequence and, so, are not random. When they are random, they are random in several distinct ways. In [72], speaking of Fourier phase values, the authors say the "distributions

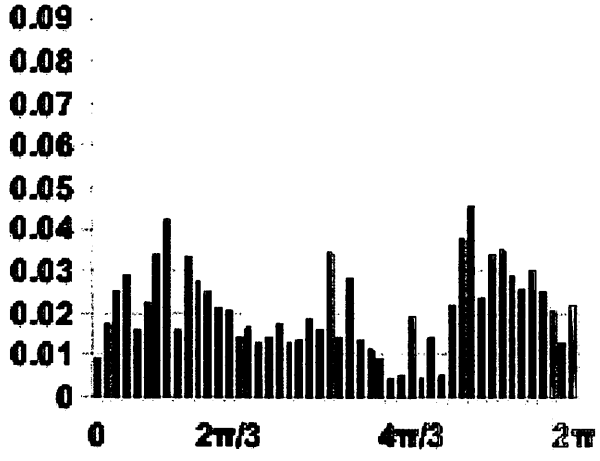


Figure 3.4: Histogram of phase values computed with a sliding window on the T sequence of the complete genome of the human T-lymphotropic virus.

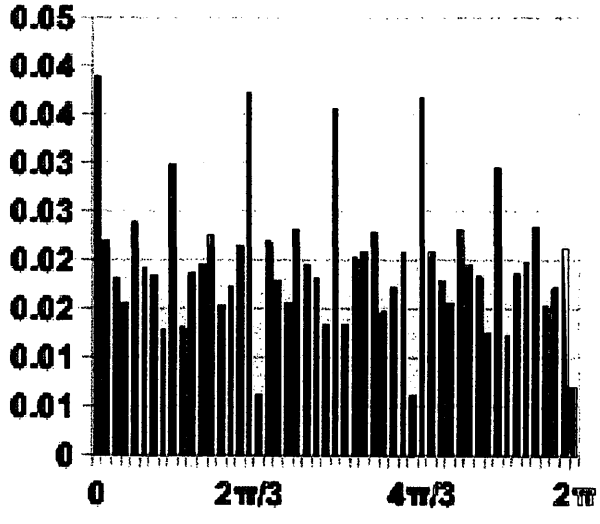


Figure 3.5: Histogram of phase values computed with a sliding window on a randomly generated binary sequence with 50% ones (reprinted from [16]).

for noncoding regions seem to be close to uniform." That statement was likely made without checking closely. In fact, these distributions have much more structure than would be expected from a uniform distribution. Notice, for example, that the histogram in Figure 2.2 divides into six regions with spikes at $0, \frac{\pi}{3}, \frac{2\pi}{3}, \pi, \frac{4\pi}{3}, \frac{5\pi}{3}$, and 2π. This pattern is commonly seen in other non-coding regions as well. It is similar to the pattern seen in the phase histogram of a completely random sequence. Figure 3.5 shows the phase histogram for a random binary sequence with an equal balance of 1s and 0s. Notice that in addition to having six spikes, the histogram repeats the same symmetric pattern three times in the intervals $\left[0, \frac{2\pi}{3}\right]$, $\left[\frac{2\pi}{3}, \frac{4\pi}{3}\right]$, and $\left[\frac{4\pi}{3}, 2\pi\right]$.

Phase histograms of random strings with different proportions of 1s and 0s maintain the same form, but the values of individual bins vary. As long as the proportions are not too skewed towards either 1s or 0s, phase distributions of distinct random sequences with the same proportions are similar. However, as Figure 3.6 demonstrates, when the proportions are skewed, the distributions can be quite different. Note that the RY/MK/SW sequence group will tend to produce sequences that are close to half and half 1s and 0s, while the A/C/G/T sequence group will tend to produce sequences that are one-fourth 1s and three-fourths 0s.

Histograms from sequences in intergenic regions sometimes look like random sequences and sometimes do not. Figure 3.7 shows two examples of Fourier phase histograms from tandem repeats, one type of sequence appearing in intergenic regions.

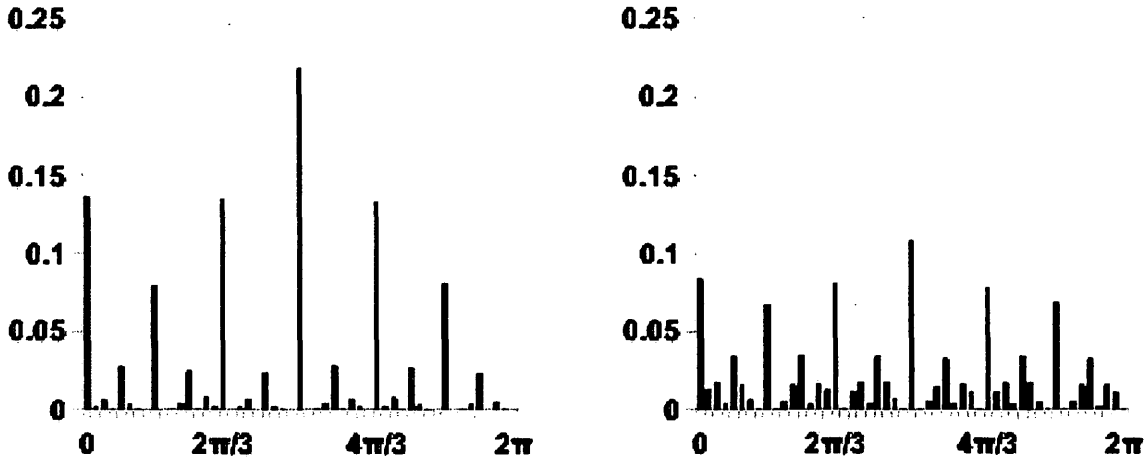


Figure 3.6: Fourier phase histograms of two different random binary sequences that are 96% ones.

Tandem repeats are regions in which sequences of two or more nucleotides are repeated many times in a row. They are used in genealogical tests. The histogram on the left resembles the histogram of a protein-coding region; the one on the right looks like a random histogram. This difference results from the length of the repeat – if it is a multiple of three, the histogram looks like the histogram of a protein-coding region.

Histograms from genes can also show aspects of randomness. This is due to the presence of introns, non-protein-coding regions, in them. Figure 3.8 shows two histograms from the same gene in human chromosome 14. The histogram on the left uses the RY sequence and resembles a histogram of a random sequence. Note the six characteristic spikes. The histogram on the right was built from the SW sequence. The grouping of windows on the right side of the histogram is a result of the exons, protein-coding regions, that code in the reading frame associated with that part of the histogram. This demon-

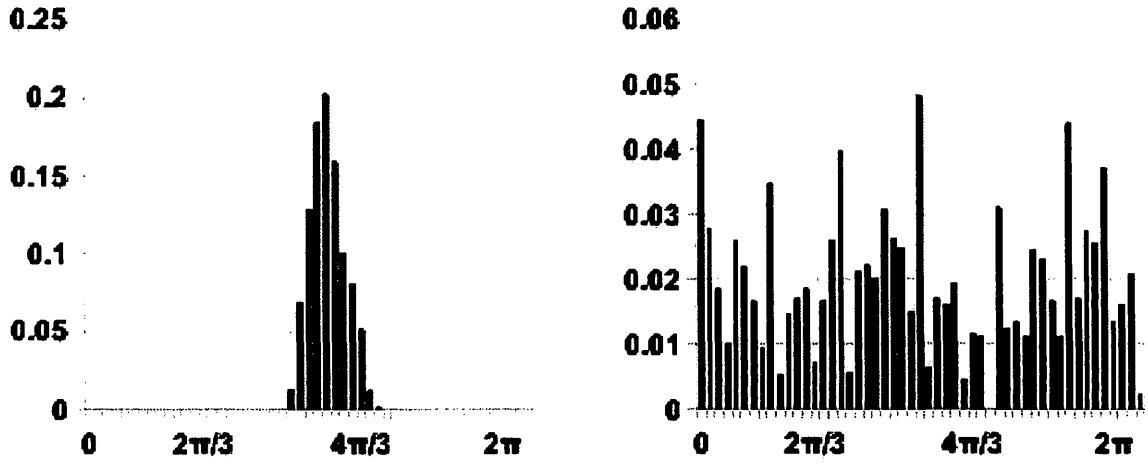


Figure 3.7: Fourier phase histograms of two different tandem repeat sequences from the human genome. Fourier phases are computed using the RY sequence.

strates the importance of combining information from multiple indicator sequences.

Fourier phase histograms can be used to investigate randomness in sequences, but not in a simple way. A Fourier phase histograph of a random sequence does not create a well-known distribution like a uniform distribution. However, random sequences do have characteristic distributions, and different types of random sequences have different characteristic distributions. There is, therefore, nuanced information about randomness available from these histograms.

3.2.2 χ^2 Features

The Pearson χ^2 goodness-of-fit test is used to create DNA sequence features by comparing the Fourier phase histogram of the sequence to the Fourier phase histogram for a random sequence with the same frequency distribution of bases. The χ^2 test is intended

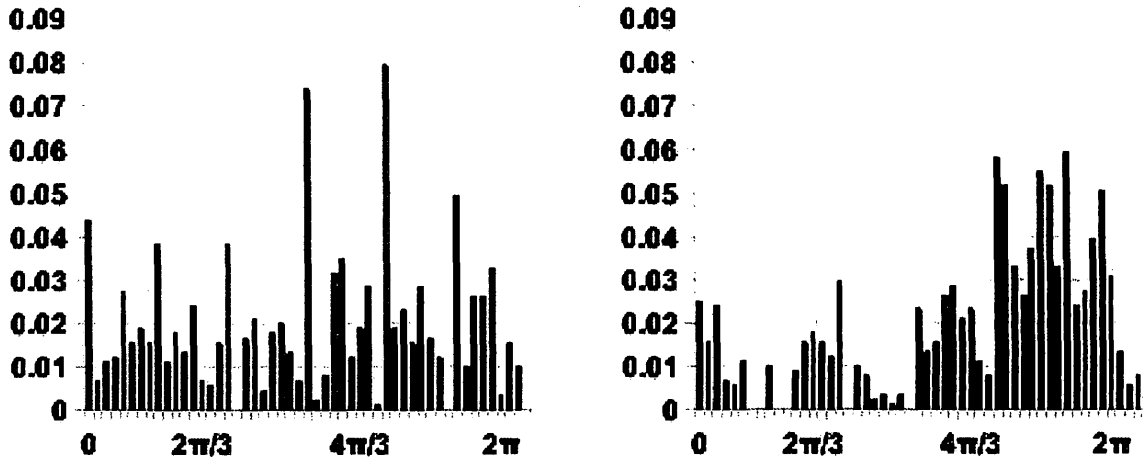


Figure 3.8: Fourier phase histograms from the same gene on human chromosome 14. The histogram on the left is built from the RY sequence; the histogram on the right is built from the SW sequence.

as a statistical test to determine how likely it is that two distributions are the same. In this work, it is instead used to produce a number that is a feature of the sequence. A DNA sequence has three χ^2 features – one each for its RY, MK, and SW binary indicator sequences.

It is calculated using the formula:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.1}$$

where O_i is the value of a bin in the Fourier phase histogram being evaluated, and E_i is the value of a bin in the Fourier phase histogram created for a random sequence, and n is the number of bins.

3.3 Sequence statistics

The following sequence statistics were used as features: the length of the sequence, the content of base types (purine, amino, strong H-bond), the length of runs of particular bases, and statistics on distances between bases.

The length of the sequence is an important feature for classifiers designed to distinguish different types of TEs. Many have characteristic lengths. Retrotransposons vary in length between 5000 and 20,000 bp, but other types of TEs have more distinctive lengths. SINEs range from 200 to 400 bp. Solitary LTRs vary in length between 100 and 3000 bp. LINEs range from 900 to 6000 bp. The analysis of IESs revealed a hitherto unknown difference in lengths, with a large group of IESs having SINE-like lengths (100-500 bp), and another large group having lengths up to 10,000 bp. It is an open research question whether this length difference is associated with a difference in origin or function.

While the values of the three divisions of bases into pairs, purine/ pyrimidine, amino/ keto, and strong/ weak H-bonds, can always be computed by combining two features from the string kernel, i.e, purine content is A-content plus G-content, it is often valuable to use such combined features in classifiers. Since these base combinations have biological meaning, it was hypothesized that these could be useful. For example, in *Tetrahymena*, it is known that coding and non-coding regions can be easily distinguished by looking at the AT-content (the value of strong/weak H-bond feature).

```
>ALU SINE1/7SL Primates
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcgggagga
ttgcttgagcccaggagttcgagaccagcctgggcaacatagcgagaccccgtctctaca
aaaaatacaaaaattagccgggcgtggtggcgcgcgcctgtagtcccagctactcgggag
gctgaggcaggaggatcgcttgagcccaggagttcgaggctgcagtgagctatgatcgcg
ccactgcactccagcctgggcgacagagcgagaccctgtctcaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
```

Figure 3.9: Sequence of the most common SINE element in humans, Alu.

The idea of looking at the length of runs of bases as a feature was inspired by SINEs. An example of a SINE is shown in Figure 3.9. Notice that it ends with what biologists call a poly-A tail, i.e., a long run of As. This distinguishes SINEs from solitary LTRs of similar length.

The final type of feature used was based on relationships between bases and what follows and precedes them. These are called *gap features*. For these the distance between pairs of dinucleotides is measured: for example the distance between GA and CC. There are 256 such pairs. Three sets of these features are used: the average distance, the maximum distance, and the minimum distance. This generates 768 features. Clearly, not all of these will be meaningful, so feature selection methods are employed to choose the meaningful ones. These feature selection methods are described in detail in Chapter 4. Some examples of effective features for distinguishing SINEs from solitary LTRs

taken from this group include: the minimum distance between GC and CG; the average distance between AA and CT; the maximum distance between AA and GG.

3.4 Conclusion

This chapter presented three sets of DNA sequence features based on signal processing and statistical properties. They will be used together with some of the features discussed in Chapter 2 and the features presented in the next chapter to identify and classify TEs.

4 Side Effect Machine Features

Side effect machines, introduced in Section 1.1.2, are finite state machines augmented with counters assigned to each state that are incremented each time the state is entered. A state in a SEM designed for DNA sequences has one transition for each base: A, C, G, and T. SEM features for a DNA sequence are calculated by running the sequence through the SEM. A DNA sequence is run through the SEM starting in State 0. The bases in the sequence control movement through the SEM. For example, suppose a sequence being run through the SEM in Figure 4.1 was in State 2 and the next base was a C. The transition to State 5 would be followed and the counter for State 5 would be incremented. The sequence would then stay in State 5 as all the State 5 transitions go to State 5, and the State 5 counter would be incremented for each subsequent base. The final values of the counters, normalized by dividing by the string length plus one, are SEM features of that DNA sequence.

SEMs are a recently developed technology for DNA sequence analysis. This thesis helps establish a solid theoretical background for SEM features by studying the genetic algorithm used to select effective SEMs for a given problem, analyzing the SEM fitness

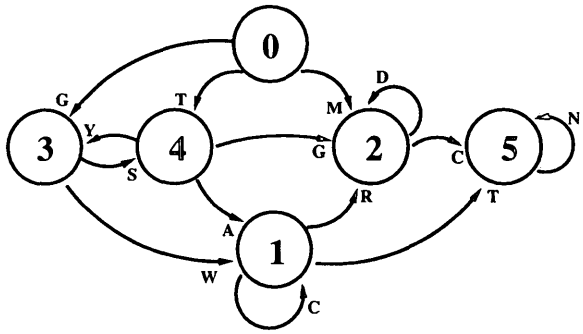


Figure 4.1: Example of an evolved 6-state SEM. Arrows are labelled with IUPAC codes (shown in Table 4.1) for DNA base transitions. States 3 and 4 form a transient communicating class. States 0, 1, and 2 are transient states, and State 5 is an attracting state. States 1 and 3 create highly effective features discussed in Section 5.2.4.1 (reprinted from [17]).

Table 4.1: IUPAC Codes for DNA bases

code	type	bases	code	type	bases
A	adenine	A	M	amino	A or C
C	cytosine	C	K	keto	G or T
G	guanine	G	H	not G	A, T, or C
T	thymine	T	B	not A	C, G, or T
R	purine	A or G	V	not T	A, C, or G
Y	pyrimidine	T or C	D	not C	A, G, or T
W	weak H-bond	A or T	N	any	A,C,G, or T
S	strong H-bond	C or G			

landscape, developing new methods for using evolved SEM features, and developing techniques for SEM feature analysis, enabling their use in Knowledge Discovery.

SEMs can operate using any alphabet, though it is best that it not be too large, since each member of the alphabet corresponds to a transition from each state. SEMs with too many transitions are difficult to analyze and easier to overtrain than SEMs with fewer transitions. For classifying DNA sequences, the natural alphabet to use is {A,C,G,T} with four transitions from each state. A 2-state SEM that calculates purine (A or G) and pyrimidine (T or C) content of a sequence using {A,C,G,T} transitions is shown in Figure 4.2. To enhance readability when drawing SEMs, multiple transitions that go to and from the same states are represented with a single arrow and, for {A,C,G,T} transitions, they are labeled with IUPAC codes (Table 4.1).

Figure 4.3 shows an example of a SEM with nine transitions instead of four. This higher-order SEM is evolved in an experiment described in Section 6.4. For that experiment, the transitions are based on reading frame data. Another obvious application for higher-order SEMs is to have transitions based on amino acids (of which there are twenty) instead of DNA bases. Using more transitions presents similar challenges to using more states. The search space becomes larger; the danger of overtraining increases; and, the SEMs become harder to interpret. The results of the experiments in Section 6.4 are not as good as the results with SEMs with four transitions. These are not insoluble problems, and future work will study these higher-order machines on more problems.

There are a large number of possible SEMs. Effective SEMs are selected using a genetic algorithm. The genetic algorithm evolves a population of SEMs using a fitness function that measures how well the features in the SEM separate the given data categories. The original SEM fitness function clustered the data using the features generated by the SEM with k-means clustering and then compared the clustering to the known division using the RAND index. The RAND index is a measure of similarity between two data clusterings and is explained in detail in Section 4.2.1.1. In subsequent research, it was found that on many problems using k nearest neighbour clustering with the RAND index produces better results. In this thesis, two new fitness functions are introduced. One is based on random forest classifiers (Section 6.1.2). Using this fitness function, the fitness of a SEM is the accuracy of a 20-tree random forest classifier built using its features. It is unusual to build a random forest with only 20 trees. The more usual number is 100 or 200. The smaller number was used because in a genetic algorithm one cannot afford the computational cost of a large number of trees. The other fitness function is based on the information theory concept of mutual information. It bases fitness on how much information the SEM-induced distance between two data objects gives about whether they are in the same or different classes.

SEMs with many states can be difficult to interpret. Previous work took the approach of evolving individual machines with enough states to produce accurate classifiers for the problem in question. In order to better understand the features used in the classifiers,

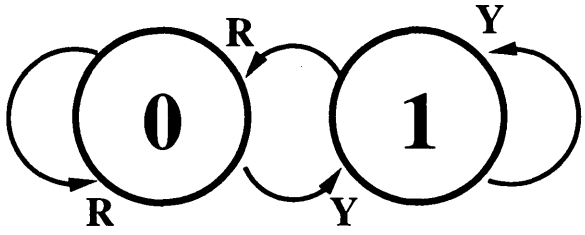


Figure 4.2: A 2-state SEM that calculates purine (R) and pyrimidine (Y) content of a sequence (reprinted from [18]). Sequence starts in State 0.

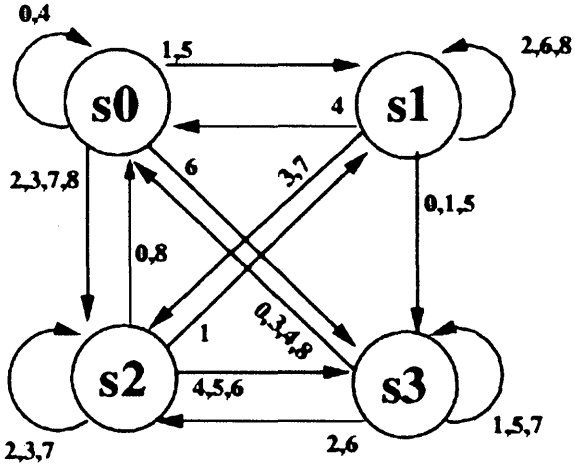


Figure 4.3: Side Effect Machine using 4 states and 9 transitions (reprinted from [14]).

an innovation is introduced in this thesis. A two-step approach is taken. First, good classifiers with a small number of states are evolved. Then, the results of many replicates are combined. Finally, an effective number of diverse features are selected from the combined set.

Some important concepts for analysis of SEMs are *transient states*, *attracting states*, and *communicating classes*. Transient states are states that, once left, are never entered again. Attracting states are states that, once entered, are never left. Communicating classes are groups of states all of which are reachable from all of the others. One can also have *transient communicating classes* and *attracting communicating classes*. These enable the SEMs to divide the sequence into modules. When a sequence is run through a SEM, it always starts in State 0.

4.1 Using Side Effect Machines

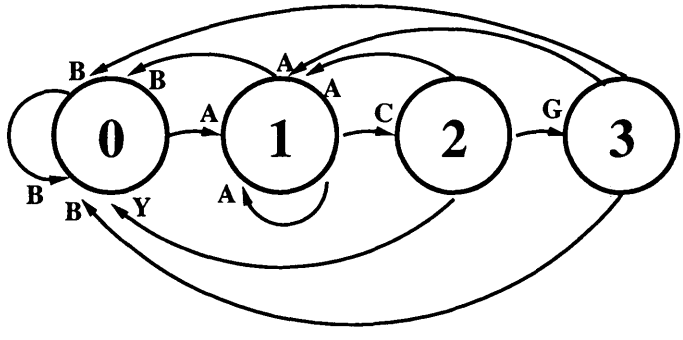


Figure 4.4: A 4-state SEM that calculates the frequency of occurrence of the 3-mer ACG using State 3. Transitions involving the bases in the 3-mer are highlighted (reprinted from [18]).

The set of all SEM features is a superset of the string kernel. Every k-mer feature can

Table 4.2: Counter values as DNA sequence "ACGACGACGACG" is run through the SEM in Figure 4.4. Columns represent the state counters. Rows represent the DNA bases. Last row is normalized counts.

	0	1	2	3
A	1	0	0	0
C	1	1	0	0
G	1	1	1	0
A	1	1	1	1
C	1	2	1	1
G	1	2	2	1
A	1	2	2	2
C	1	3	2	2
G	1	3	3	2
A	1	3	3	3
C	1	4	3	3
G	1	4	4	4
	0.08	0.31	0.31	0.31

be generated using a SEM with $k + 1$ states. Figure 4.4 shows a SEM that computes the k-mer "ACG" using State 3. Suppose, for example, the DNA sequence, "ACGACGACGACG" (4 repeats of ACG) is passed through this SEM. Table 4.2 shows the values the counters would have. State 1 is counting the number of As; State 2 is counting the number of ACs; State 3 is counting the number of ACGs. State 0 only counts the beginning of the sequence in this example, but, in a different sequence, it would count all bases that were not a part of ACG subsequences.

Tables 4.3 and 4.4 were created to provide an example of what SEM and k-mer features can measure. These tables contain feature values for the SINE Alu sequence (Figure 3.9). Table 4.3 shows the features generated by the SEM shown in Figure 4.1, and Table 4.4 shows the 2-mer features for the same sequence. Note that the value of

Table 4.3: SEM feature counts and values for the SINE Alu sequence shown in Figure 3.9 using the 6-state SEM shown in Figure 4.1.

state	0	1	2	3	4	5
count	1	0	3	2	2	304
value	0.003	0.000	0.010	0.006	0.006	0.974

Table 4.4: 2-mer features for the SINE Alu sequence shown in Figure 3.9.

AA	CA	GA	TA	AC	CC	GC	TC
0.129	0.058	0.074	0.026	0.039	0.074	0.106	0.042

AG	CG	GG	TG	AT	CT	GT	TT
0.090	0.068	0.093	0.055	0.026	0.061	0.035	0.026

SEM feature 5 is much larger than the values of the other features. This is because it forms its own communicating class. Once State 5 is entered, it is never left. Also note that the value of SEM feature 1 is zero. This sequence never enters State 1. Table 4.4 shows the 2-mer features for this sequence. There are 16 2-mers. Note that the AA feature has a relatively high value. This is because of the poly-A tail at the end of Alu. GC is common in this sequence, but TA, AT, and TT are rarer.

As well as k-mers, the SEM structure allows for the representation of a broad variety of other patterns in the sequence. In [67, 68] motifs, features that are short sequences represented using IUPAC codes (Table 4.1), are shown to improve performance over k-mer features. SEMs can represent these as well.

SEMs can generate even more types of features. SEMs count the frequency of occurrence of patterns based on regular expressions. As well as representing patterns that can be built using wild cards, the patterns can depend on what came before or what did

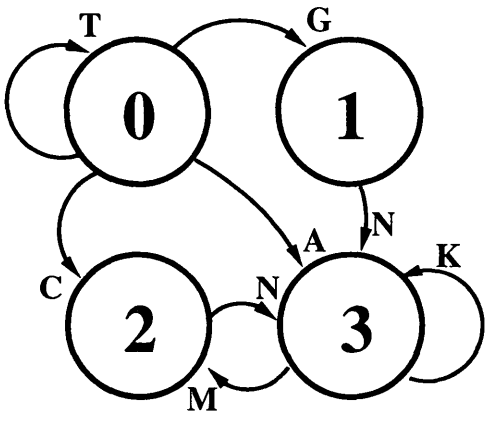


Figure 4.5: A 4-state SEM, evolved using sLTR data, with multiple communicating classes (reprinted from [18]).

not come before and can include variable size repeats or variable size gaps and those gaps can be restricted to specific patterns. Each of the n states in a SEM measures the frequency of occurrence of a pattern, or, more formally, the frequency of occurrence of strings of a regular language defined by the SEM finite state machine with that state as the sole accepting state.

SEMs can divide the sequence into modules, by using multiple communicating classes in their finite state machine. This division enables them to find features in particular portions of the sequence, such as features of the initial portion of the sequence. An example of this kind of SEM is shown in Figure 4.5. States 0 and 1 are transient (once left never returned to) and calculate features of the starting portion of the sequence. State 0 calculates the proportion of the sequence consisting of initial Ts; State 1 is zero if the sequence does not start with T^*G, non-zero otherwise. States 2 and 3 calculate the amino/keto content of the rest of the sequence (ignoring repeated amino bases).

Some examples of features easily modelled by SEMs: purine content, frequency of occurrence of As that follow sequences of the form CT^*C, frequency of occurrence of runs of Ts.

SEMs are reminiscent of profile hidden Markov models (pHMMs) [43], which are also built with finite state machines (though they use probabilistic rather than deterministic finite state machines). Instead of generating sequence features, pHMMs build a model of the sequence. The test sequence is run through the model from beginning to end. To increase the scores in a pHMM, a subsequence must both fit a pattern and be in the right place in the sequence. SEMs have a beginning (State 0), but no end. They count the number of occurrences of subsequences corresponding to patterns without regard to their position. SEMs are computationally simpler than pHMMs. For scoring, pHMMS have time complexity $O(NM)$ (where N is the length of the sequence and M is the number of states), while computing SEM features has time complexity $O(N)$. In order to be effective, pHMMs require that good multiple sequence alignments exist for the sequences being classified. As a rule of thumb, when sequences are better recognized through local correlations, SEMs will produce better classifiers.

SEMs are more general versions of motif features based on regular expressions, since any finite state machine can be represented as a regular expression [120]. SEMs measure the presence or absence of a subsequence matching a particular regular expression. SEMs measure the frequency of occurrence of these subsequences. It is conceivable that more

expressive languages work better than regular expressions for regular expression motif features. This is because regular expressions are not robust to insertions and deletions, and insertions and deletions are common in biological sequences. The counters in SEMs protect them from this problem. While SEMs occasionally use transient states to detect the presence or absence of a regular expression motif, SEM states usually count members of a regular language that occur many times in the sequence. For these, a single insertion or deletion does not substantially disrupt their counts.

The number of SEMs grows super-exponentially with the number of states. There are n^{tn} n-state SEMs with t transitions. This means there are more than 4 billion 4-state 4-transition SEMs, each of which contains 4 features. For a given problem, there are many diverse useful SEM features. This creates the possibility of many accurate classifiers with different sets of SEM features.

Since the set of all SEM features is so large, it is necessary to use some mechanism for finding effective ones for the problem at hand. A genetic algorithm is used to do this. The genetic algorithm requires training data with known classes and evolves n-state SEMs whose n features create good classifiers for that data. In the original SEM research, these n features were used as an end product. In this thesis, an innovation was introduced in which the best SEMs from multiple replicates of the genetic algorithm are saved, their features pooled, and feature selection used to select a set of good features for the problem. This allows the choice of features that are more comprehensible than those

used in the original method, since the SEMs have a smaller number of states, while still producing a high accuracy classifier.

4.2 Genetic Algorithm

A genetic algorithm is a population-based optimization technique. It is inspired by the biological theory of evolution. The structure of the genetic algorithm that is used to evolve SEMs is shown in Algorithm 2. This is a standard steady state genetic algorithm using double tournament selection.

Algorithm 2: SEM Genetic Algorithm

Data: Training data with labels D, fitness function f, population size n, tournament size ts, stop condition S
Result: SEM
Initialize a population of size n of SEMs
Choose a subset of D at random for fitness assessment
Assess fitness of all SEMs in initial population using f
while S *not met* **do**
    Choose ts SEMs from the population at random (a tournament)
    Pick the two most fit from the tournament to be parents
    Apply crossover operator to create two children from the parents
    Apply the mutation operator to the children
    Assess fitness of the children
    Replace the two least fit from the tournament with the children
end
return SEM with highest fitness

In order to understand Algorithm 2, it is necessary to know how the SEMs are represented, the crossover operator, the mutation operator, and the fitness function. SEMs are represented in the genetic algorithm as an $n \times t$ array with each row representing one of

State	A	C	G	T
0	3	2	1	0
1	3	3	3	3
2	3	3	3	3
3	2	2	3	3

Figure 4.6: Representation of SEM shown in Figure 4.5 used in the genetic algorithm (reprinted from [17]). A number in the matrix is the state transitioned to from the state in its row upon encountering the base in its column.

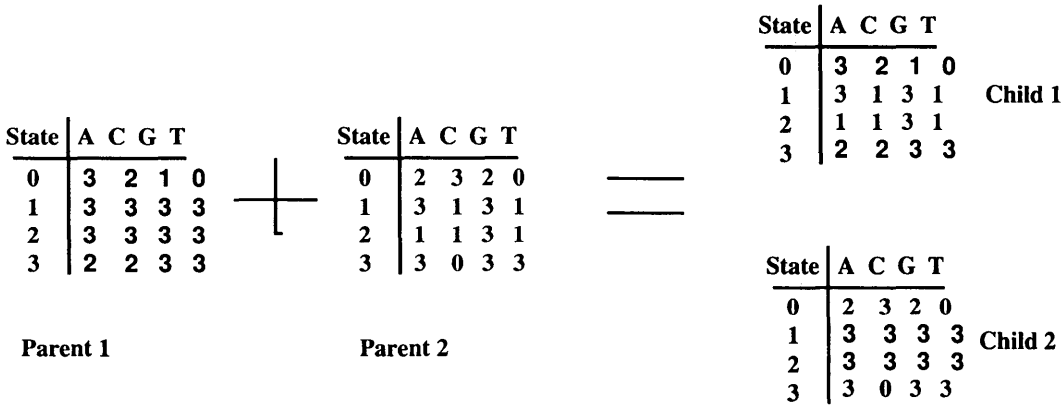


Figure 4.7: Example of crossover in 4-state SEMs.

the n states and each column one of the t transitions to other states. Figure 4.6 shows an example. In this example, there are four states and four transitions, one for each of the bases, A, C, G, and T.

Two-point crossover on the vector of states is used. This means two numbers are picked at random, p_1 and p_2 in $(1, n)$, where n is the number of states, and the rows are exchanged in $[p_1, p_2]$ in the parents to create the children. An example of this on 4-state SEMs is shown in Figure 4.7. Point mutation is used, which picks an array element at random and changes its value to another valid value.

The genetic algorithm is performed off-line so computational time can be and, in fact, is long. The number of mating events needed to get good results is determined by doing a few preliminary tests. For the experiments in this thesis, it is found that between 2000 and 6000 mating events are needed. This is a small number. Many genetic algorithms run for many more mating events. Each replicate of the algorithm takes 5-10 minutes to complete.

4.2.1 Fitness Functions

The use of four different fitness functions for the SEM genetic algorithm are investigated. The first two, k-means fitness and k nearest neighbour fitness have been used in previous work. Two others are introduced: random forest fitness and information gain fitness.

4.2.1.1 K-means Fitness Function

The *k-means fitness function* was used in the original SEM research [13, 10]. With this fitness function, the fitness of a SEM is calculated by clustering the data using k-means clustering [86] based on the n features in the n-state SEM on a set of training data and then computing the RAND index with the division based on the known classes. The fitness is the value of the RAND index.

The RAND index is a method for comparing two clusterings. It is defined as follows [110]:

Definition 1 *Let X be the set of N objects to be clustered, $\{X_1, X_2, \ldots, X_N\}$.*

Let Y be a specific partitioning of X into K disjoint sets (a clustering).

Write Y as a set of clusters $Y = \{Y_1, Y_2, \ldots, Y_K\}$ where each cluster is a set of the given points $Y_k = \{X_{k_1}, X_{k_2}, \ldots, X_{k_{n_k}}\}$ with $\sum_k n_k = N$ and $n_k \geq 1$ for $k = 1, 2, \ldots, K$.

Let Y' be another clustering of X into K clusters.

Let n_{ij} be the number of points simultaneously in Y_i and Y'_j.

Then the **RAND index** *of Y and Y' =*

$$\frac{\binom{N}{2} - \left(\frac{1}{2}\left(\sum_i(\sum_j n_{ij})^2 + \sum_j(\sum_i n_{ij})^2\right) - \sum_i\sum_j n_{ij}^2\right)}{\binom{N}{2}} \tag{4.1}$$

In plain language, the RAND index is the proportion of pairs of points that are either both in the same cluster in the two clusterings or both in different clusters in the two clusterings. It is a number between zero and one. Therefore, a fitness of one means that k-means clustering divides the data into exactly the groups designated by the training labels.

4.2.1.2 K Nearest Neighbour Fitness Function

A similar fitness function, the *k nearest neighbour fitness function* was designed to work with k nearest neighbour clustering instead of k-means clustering. For this fitness function, it is required that a subset of the training data be designated as neighbours. The features created by the SEM make it possible to determine the distance between data points. Thus each point in the test set can be given a class designation based on its k nearest neighbours. The set of neighbours and the test set are changed at intervals during

evolution in order to avoid overfitting. And, again, the fitness value is the value of the RAND index comparing the k nearest neighbour clustering with the known clustering.

4.2.1.3 Random Forest Fitness Function

The *random forest fitness function* is a new fitness function for SEMs. The random forest fitness is the out-of-the-bag (OOB) error of a random forest created using the SEM and the training data. A smaller number of trees than usual is used (20 instead of the more usual 100 or 500) to reduce the time needed to build the random forest. For the purpose of measuring fitness this is acceptable, because all that is needed is a way to compare two different feature sets, not an optimal classifier.

4.2.1.4 Information Gain Fitness Function

Another fitness function introduced here is the *information gain (IG) fitness function.* Information gain is measured using mutual information. Mutual information is a measure of how much information one random variable X gives about another random variable Y. In the case of this fitness function, X is a set of bins with the distances between pairs of members of the training data set based on the SEM features created by a particular SEM, and Y is information about whether each pair has the same label or different labels. Mutual information (MI) is defined as:

$$H(X) + H(Y) - H(X, Y) \tag{4.2}$$

$H(X)$ and $H(Y)$ are entropies defined by

$$H(X) = -\sum_{x \in X} P(x) \ln P(x) \tag{4.3}$$

where $P(x)$ is the frequency of occurrence of x in X. $H(X,Y)$ is the joint entropy of X and Y and is defined by:

$$H(x,y) = -\sum_{x \in X} \sum_{y \in Y} P(x,y) \ln P(x,y) \tag{4.4}$$

where $P(x,y)$ is the frequency of x and y occurring together. In this case, $Y = \{0,1\}$ with $y = 0$ meaning that a pair of data elements has two different labels, and $y = 1$ meaning that a pair of data elements has two identical labels. $P(x,0)$ is thus the frequency that a pair of data elements separated by a distance in bin x has different labels. Algorithm 3 shows how the IG fitness function is calculated.

Algorithm 3: IG Fitness Function

input : $S \leftarrow$ SEM, $D \leftarrow$ training data with labels, $H_Y \leftarrow$ entropy of pairs in D based on labels, $BIN \leftarrow$ number of bins
output: fitness

Calculate features for D using S
$DIST \leftarrow$ the distances between all pairs in D calculated with the SEM features
$SAME \leftarrow$ the distances between all pairs in D with identical labels
$DIF \leftarrow$ the distances between all pairs in D with different labels
$HIST \leftarrow$ histogram for $DIST$ with BIN bins of equal width
$HIST_S \leftarrow$ histogram for $SAME$ with BIN bins of equal width
$HIST_D \leftarrow$ histogram for DIF with BIN bins of equal width
$H_X \leftarrow$ entropy of $HIST$
$H_{X,Y} \leftarrow$ joint entropy calculated from $HIST_S$ and $HIST_D$
$IG \leftarrow H_X + H_Y - H_{X,Y}$
return IG

These four fitness functions, k-means fitness, k nearest neighbour fitness, random forest fitness, and IG fitness, create distinct fitness landscapes that direct the search of the genetic algorithm differently. This results in different features located. Therefore, it is important to study the fitness landscape.

4.3 SEM Fitness Landscape

A fitness landscape is defined to be the combinatorial graph with each possible solution as a vertex, edges connecting vertices that differ by a distance defined appropriately for the problem, and a fitness value assigned to each vertex. A SEM fitness landscape will depend on: the number of states in the SEM, the fitness function used, the data set used by the fitness function, and some sort of distance measure between the SEMs. SEMs with the number of states $n = 4$ are analyzed. This choice of n produces effective and comprehensible SEM features. The analysis is started using the random forest fitness function, and later aspects of the fitness landscapes of all four fitness functions are compared. The analysis is done based on three different data sets:

- **sLTR/SINE problem** This data set uses solitary LTR sequences extracted from ERVs found by Retrotector and SINEs identified by RepeatMasker. Only RepeatMasker sequences that are identified as complete matches are used. This data set has 289 solitary LTR sequences and 499 SINEs. The sequences are of average length 389 bases. Sequences identified by Retrotector were chosen because they

are the best representatives of ERVs in the human genome available, since they are identified based on biological characteristics instead of just sequence homology. In Chapter 5 the sLTR/SINE problem is explored with different data sets.

- **RT problem** This data set uses three types of sequence from the *Drosophila melanogaster* genome: 104 complete LTR retrotransposons, 118 exons, and 186 intergenic sequences. These were collected using annotations in FlyBase [134]. The average length of these sequences is 7468 bases.

- **IES problem** 218 IES and 222 MDS sequences were extracted using BLAST with sequences taken from the Tetrahymena Comparative Sequencing Project. The MIC genome was blasted against the MAC genome and common sequences were designated as MDSs, while sequences found only in the MIC were designated as IESs. The average length of these sequences is 5609 bases.

These data sets were chosen because they have both commonalities and differences. They all involve transposable elements, so there should be sequence features in common. They come from different organisms. The RT problem is a three-way classification; the others are two-way classifications. The IES problem involves heavily mutated sequences. The sLTR/SINE problem distinguishes two types of transposable elements, while the other two problems distinguish transposable elements from other types of sequences. The solitary LTRs and SINEs are much shorter than the sequences in the other data

sets and have been shown to have modular structures [20, 42] involving differences in sequence composition.

4.3.1 Genotypic Fitness Landscape

The natural distance measure to use to analyze the fitness landscape is the variation operator used during evolution, in this case crossover together with mutation. However, the crossover-based fitness landscape is intractable, since its analysis requires computing for each pair of SEMs which SEMs can be reached with one crossover operation. It is easier, and still useful, to analyze the mutation fitness landscape created by connecting vertices whose SEMs differ by one mutation.

For n-state SEMs the fitness landscape graph is always the same, but the fitness values vary depending on what training data is used. Fitness landscapes are often described in analogy to a natural landscape with hills and valleys. The graph represents the ground and the fitness its height. This analogy does not work well for SEM fitness landscapes. A single mutation can cause a large change in fitness, in either direction, so there is not a smooth gradient in analogy to a hillside. Since there are more than 4 billion 4-state SEMs, it is not possible to completely describe or visualize the fitness landscape. It is only possible to analyze crucial properties of it.

Figures 4.8-4.10 show visualizations of a portion of the fitness landscapes for the three data sets. The figures all use the same randomly generated 500 SEMs, but are

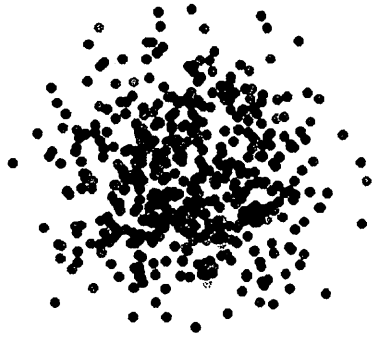


Figure 4.8: Visualization of a portion of the fitness landscape for the IES problem based on 500 randomly generated 4-state SEMs. Darker circles have better random forest fitness (reprinted from [18]).

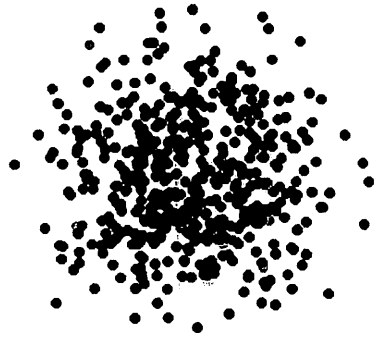


Figure 4.9: Visualization of a portion of the fitness landscape for the RT problem based on 500 randomly generated 4-state SEMs. Darker circles have better random forest fitness (reprinted from [18]).

coloured differently based on the fitness values for the three different problems. The points are spaced based on a multi-dimensional scaling calculated from their mutation distance matrix, where mutation distance is the minimum number of mutations needed to get from one SEM to the other. Multi-dimensional scaling allows the visualization of a multi-dimensional space in two dimensions. The SEMs represented by dots are the same in all three figures, but are coloured differently based on their fitness for that problem. Note that, in all cases, low and high fitness SEMs appear close together. Also, note that the highest fitness points are scattered throughout the space and are different for the different problems. They are non-existent in the IES sample, common in the RT sample, and sparse in the sLTR/SINE sample.

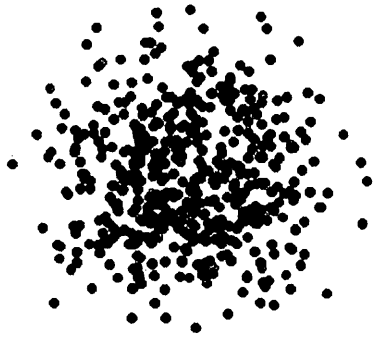


Figure 4.10: Visualization of a portion of the fitness landscape for the sLTR/SINE problem based on 500 randomly generated 4-state SEMs. Darker circles have better random forest fitness (reprinted from [18]).

Figures 4.8-4.10 show the fitness landscapes for randomly selected SEMs. For SEMs found by evolution, the neighbourhoods (SEMs one mutation away) differ in character by problem. The SEMs evolved to solve the IES problem have the sharpest "peaks." Twenty-two percent of the evolved SEMs have no neighbours with the same or higher fitness, and the median value is 4% of neighbours have the same or higher fitness. Almost as many of the SEMs evolved to solve the RT problem have no neighbours with the same or same or higher fitness (19%), but these SEMs have a higher median proportion of same or higher fitness neighbours, 8%. The SEMs evolved to solve the sLTR/SINE problem have the flattest "hilltops", with a median proportion of 16% same or higher fitness neighbours, and only 6% of SEMs having no same or higher fitness neighbours. For all the problems, there were a few SEMs found that had more than 40% same or

higher fitness neighbours. This analysis suggests that the IES optima are the hardest of the three to find, and that the sLTR/SINE optima are the easiest.

4.3.2 Comparison Of Genetic Algorithm To Random Search

For many problems to which SEMs have been applied, including these, the initial population has good average fitness. This means that "sea level" in the fitness landscape is pretty high. This explains why relatively few mating events are needed in the genetic algorithm – there is not far to climb. Figure 4.11 shows the distribution of random forest fitnesses for the three problems studied for 10,000 random 4-state SEMs. Also shown are the fitnesses of SEMs found by evolution. This figure shows that random search finds good features, but that evolution finds better ones. Excellent SEMs are rarer than one in ten thousand, i.e. there are less than 400,000 of them. Most 4-state SEMs have fitnesses that vary only through a range of less than $\pm 5\%$ accuracy with a small proportion being exceptional and a small proportion being useless.

4.3.3 Comparison Of Genetic Algorithm To Greedy Hillclimber

As an alternative to the genetic algorithm, a greedy hillclimber with random restart was tried. The greedy hillclimber selects a SEM at random, examines all its neighbours one mutation away, selects the one with best fitness (if better than its own fitness), and repeats the process with that SEM, continuing to climb the hill until none of its neighbours have

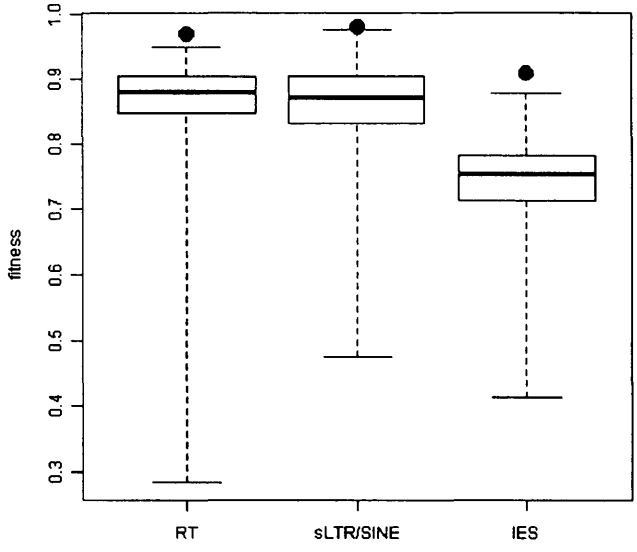


Figure 4.11: Distribution of random forest fitness for random selection of 10,000 machines. Dots represent the fitness of machines found by evolution (reprinted from [18]).

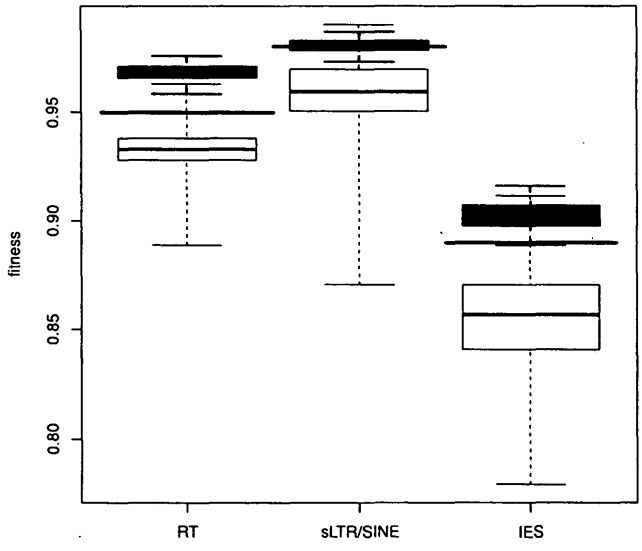


Figure 4.12: Comparison of distributions of fitnesses for SEMs found by the genetic algorithm and SEMs found by the greedy hillclimber using the same number of fitness evaluations. Genetic algorithm distribution is shown with the filled red boxplots. Greedy hillclimber distribution is shown with the open black boxplots with the blue horizontal line indicating the cutoff for the best 100 SEMs produced by the hillclimber.

better fitness than it does. The current SEM is saved and its fitness is recorded. Then, another random SEM is chosen. This process continues until a specified number of fitness evaluations have been done. In this case, the number of fitness evaluations specified was the same as the number used in one hundred replicates of the genetic algorithm. Thus, both algorithms were run for the same amount of time (many hours).

Figure 4.12 shows the results compared to those of the genetic algorithm. While the genetic algorithm produced 100 SEMs, the greedy hillclimber produced 1358 SEMs for the sLTR/SINE problem, 6200 for the RT problem, and 5212 for the IES problem. So, for the purpose of finding quality features, only the best 100 produced by the greedy hillclimber should be considered. The blue lines in the figure indicate the worst fitness for these groups. The figure shows that this method produces SEM features of similar quality to those produced by the genetic algorithm. It is not done in this thesis, but it would be reasonable to use these instead of the features generated by the genetic algorithm or to pool both sorts of features. The genetic algorithm is more likely to find features with small basins of attraction, while the greedy hillclimber will find features on broader hilltops, so, for many problems, the algorithms will find different SEM features.

The hillclimber also gives insight into the shape of the fitness landscape. For all three problems, it found SEMs exhibiting a wide range of fitness values. This indicates that the hills in the landscape have a variety of heights. For all three problems, the low fitness random SEMs proved to be connected to higher fitness SEMs. The larger number of

SEMs found by the hillclimber for the RT problem indicates that it has many short hills. The number of upward steps for this problem ranged from 1 to 10 with an average of 4.0. The hillclimber found a smaller number of hills for the sLTR/SINE problem in part because it is easier to find high quality SEMs for this problem and so both the genetic algorithm and the hillclimber were run for as one-third as long. Even so, the results of the hillclimber indicate that its hills are taller with the number of upward steps ranging from 2 to 14 with an average of 6.1. This suggests that this landscape has fewer taller hills. The IES landscape seems to have both short and tall hills with the number of upward steps ranging from 1 to 14 with an average of 4.8.

4.3.4 Phenotypic Fitness Landscape

Mutation distance gives one a sense of how evolution explores the fitness landscape. However, it is possible for two SEMs that create identical features to be far apart in that fitness landscape. Consider two SEMs with the same State 0 and the other states identical except for their numbering. These two SEMs produce the same features, but can be many mutations apart. Another way to create a fitness landscape is using a distance measure based on the difference in behaviour of the generated features. This is called a *phenotypic fitness landscape*. To distinguish the two fitness landscapes, the fitness landscape based on mutation distance is referred to as the *genotypic fitness landscape*.

The phenotypic fitness landscape is created using correlation distance. Correlation

distance is $1 - |r|$, where r is the Pearson correlation between two SEM features. Correlation distance varies from zero to one.

Definition 2 Pearson correlation *for samples* $X = \{x_1, x_2, \ldots, x_m\}$ *and* $Y = \{y_1, y_2, \ldots, y_m\}$ *with means* $\overline{x}$ *and* $\overline{y}$*, respectively, is*

$$r = \frac{\sum_{i=1}^{m}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{m}(y_i - \overline{y})^2}} \tag{4.5}$$

The phenotypic fitness landscape gives information about the diversity of the features found. While the density of optima in the genotypic fitness landscape determines the difficulty of search, the density of evolved features in the phenotypic fitness landscape determines how many functionally different features have been found. Note that this fitness landscape is based on SEM features, not complete SEMs. This landscape includes n features for every n-state SEM.

Figure 4.13 shows the number of near neighbours for each of the evolved features in this fitness landscape. Near neighbours are other evolved features within a distance of 0.2. These are displayed in sorted order. For all three problems, some evolved features have few other evolved features close to them. The RT problem has a large group, containing about half the evolved features, that are close to more than a third of the other evolved features. Slightly more than half of the sLTR/SINE features have less than 30 near neighbours, while the others are close to as many as a fourth of the other features. The IES features have a range of densities, with some features having no near neighbours and others having 28% of the other features as near neighbours.

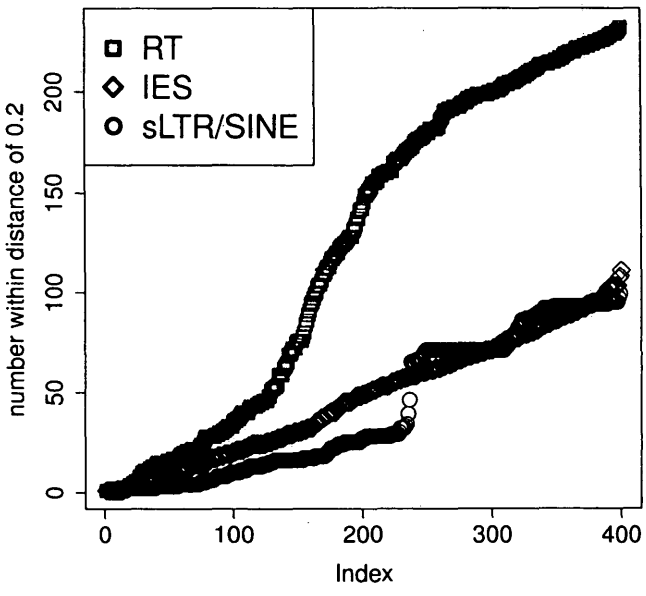


Figure 4.13: Number of other evolved features within a distance of 0.2 of 400 evolved features in the phenotypic fitness landscape. Features sorted based on increasing number of near neighbours (reprinted from [18]).

The phenotypic fitness landscape is examined for the string kernel with $k = 1 \ldots 4$ for comparison. There are 4^k k-mers. So, for $k = 1 \ldots 4$ there are 340 k-mer features. String kernel features are packed much less densely in the space. Most (58% for the RT problem; 84% for the IES and sLTR/SINE problems) have no other string kernel features within a distance of 0.2. Even those which do have other string kernel features nearby have few of them – a maximum of 24 for the RT problem and maximums of 6 for the IES and sLTR/SINE problems.

Figures 4.14-4.16 show multi-dimensional scalings of the portion of the phenotypic fitness landscape occupied by the evolved features of each problem together with the

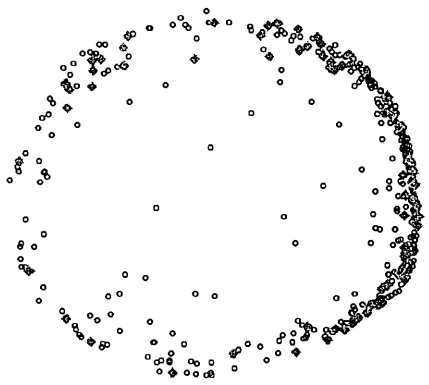


Figure 4.14: Visualization of portion of the phenotypic fitness landscape for the RT problem. SEM features are shown as black circles; string kernel features as grey diamonds (reprinted from [18]).

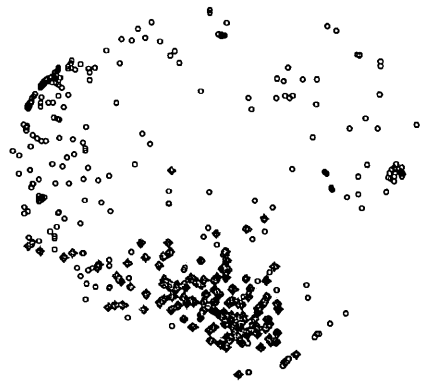


Figure 4.15: Visualization of portion of the phenotypic fitness landscape for the sLTR/SINE problem. SEM features are shown as black circles; string kernel features as grey diamonds (reprinted from [18]).

string kernel features that have the best mutual information scores with the sequence classification, i.e. those that are most useful for the problem. These figures provide intuition into how diverse the evolved features are and how similar they are to the k-mer features.

Figure 4.14 shows the evolved and k-mer features for the RT problem. A large group of features that are similar to each other are on the right side of the figure. Of the three problems studied, this one distributes the k-mer and evolved features together the most. About a quarter of the k-mer features (28%) have evolved features as near neighbours (within a distance of 0.2), and some have as many as half the evolved features as near

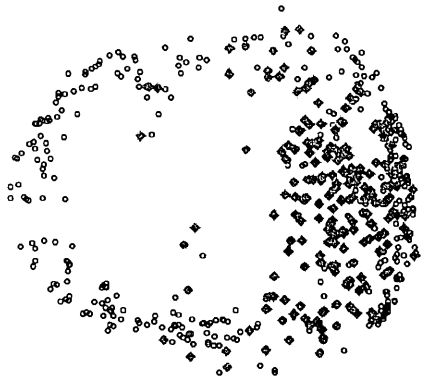


Figure 4.16: Visualization of portion of the phenotypic fitness landscape for the IES problem. SEM features are shown as black circles; string kernel features as grey diamonds (reprinted from [18]).

neighbours. 90% of the SEM features have k-mer features as near neighbours, though a small group, average size 12.

Figure 4.15 shows the evolved and k-mer features for the sLTR/SINE problem. The evolved features for this problem form groups of varying sizes with scattered isolated features. The k-mer features are distributed differently from the SEM features. Most k-mer features (97%) have no SEM features as near neighbours, and most SEM features (92%) have no k-mer features as near neighbours.

Figure 4.16 shows the evolved and k-mer features for the IES problem. The SEM features can be roughly divided into three groups – the largest at the far right, one in the upper left, and the other in the lower left. The k-mer features are closest to the group at

the far right. As in the sLTR/SINE problem, the k-mer and SEM features are differently distributed. Most k-mers (91%) have no SEM features as near neighbours. However, unlike in the sLTR/SINE landscape, a few have as many as 100 SEM feature near neighbours. Most SEM features (62%) have a few k-mer near neighbours (maximum group size 7).

Figure 4.17 shows a visualization of the portion of the phenotypic fitness landscape containing all the evolved features for the three problems. Correlation distance was calculated using the three data sets combined. In this figure, it appears that the evolved SEMs for the IES problem and the RT problem are similar, while the evolved SEMs for the sLTR/SINE problem are different. The similarity supports the belief held by biologists that IESs are mutated retrotransposons. Further evidence for this can be found in Section 4.3.6. It makes sense that useful features for the sLTR/SINE problem would be different, because, those features need to distinguish between two different types of transposons instead of between transposons and non-transposons.

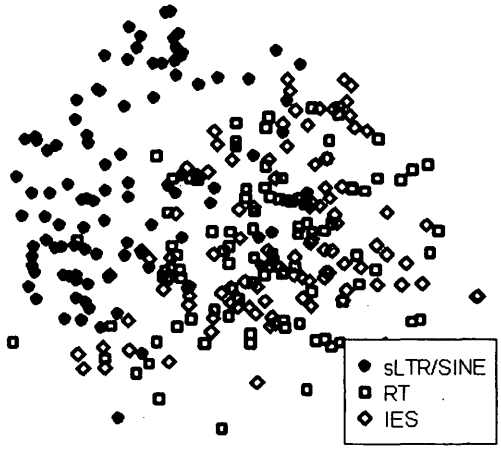


Figure 4.17: Projection onto two dimensions of machines found by evolution. Circles represent machines evolved to solve the LTR problem; squares the RT problem; diamonds the IES problem (reprinted from [18]).

4.3.5 Evolving With Different Fitness Functions

The IES problem was used to test the impact of changing the fitness function on the features found. Figure 4.18 shows a visualization in the phenotypic fitness landscape of the features from the best SEMs found by each fitness function for 25 replicates. The different fitness functions explore the search space differently. The random forest (rF) fitness function and the k nearest neighbour (knn) fitness function find SEM features covering a larger range of the landscape than the other fitness functions do. Perhaps this is because they are stochastic fitness functions. Because they train using subsets of the training data, they are able to find a greater range of features. The information gain (IG)

and k-means fitness functions find a similar range of features, which are in the same part of the fitness landscape for which the SEM features evolved with the rF and knn fitness functions are most dense. Outside the region that contains the IG and k-means features, the rF and knn fitness functions find features far apart from each other.

Because all these fitness functions are based on classifiers and classifiers sometimes overfit the data, the questions arise of whether that is happening during the course of evolution and of how that effects the results of the genetic algorithm. Remember that it is features that are evolved, not classifiers. Features are harder to over fit than classifiers are. An overfitted classifier in the fitness function might misdirect evolution, but would not necessarily produce an overfitted feature. Of the four classifiers used in the fitness functions, random forests are least likely to over fit. The knn fitness function changes the data points used as neighbours periodically, a quality that should reduce overfitting. The IG and k-means fitness functions have no design features to prevent overfitting. However, Figure 4.18 suggests that the features found using the IG and k-means fitness functions are mostly similar to features found by the knn and random forest fitness functions. So, for this problem, overfitting of the fitness function does not seem to be a problem.

Figure 4.19 shows a comparison of the quality of features found by the different fitness functions. The measure of quality is information gain. In this case, information gain is calculated for individual features using the feature values, as opposed to how it is used in the fitness function for n-state SEMs. Features found using the IG and

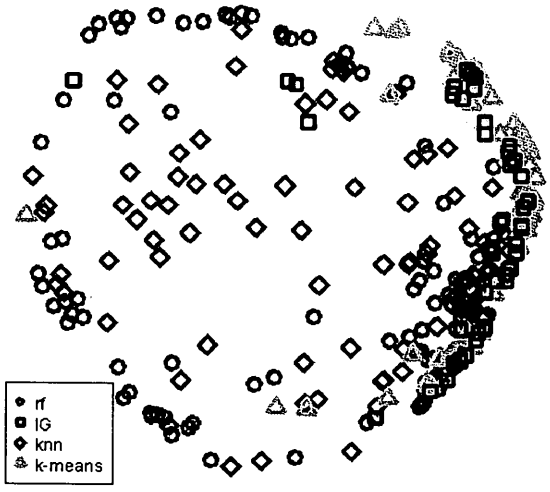


Figure 4.18: Projection onto two dimensions of machines found by different fitness functions. Circles represent SEMs evolved using the random forest fitness function; squares the IG fitness function; diamonds the knn fitness function; triangles the k-means fitness function (reprinted from [18]).

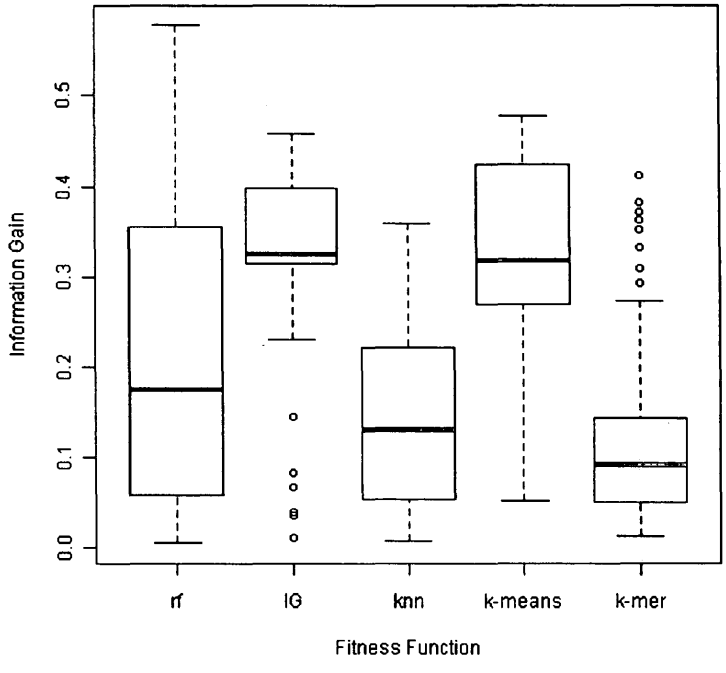


Figure 4.19: Distributions of information gain for features evolved with various fitness functions and for k-mer features. Not shown are outliers of the k-mer features with negative information gain (reprinted from [18]).

k-means fitness functions have the highest mean information gain, but the feature with the maximum information gain was found using the rF fitness function. The features found using the knn fitness function have both the lowest mean and the lowest maximum information gain.

However, the fact that the individual features are more informative does not mean that the end product classifier will be better. Table 4.5 shows the error on test data for classifiers built from the SEM features generated by the various fitness functions. The

Table 4.5: Error on test data for random forests trained using various feature sets.

fitness function	all	rF	IG	DC
rF	8%	8%	8%	9%
IG	11%	12%	17%	12%
knn	13%	14%	13%	13%
k-means	11%	12%	15%	11%

features produced by the rF fitness function produce the most accurate classifiers for this problem, and those produced by the knn fitness function the worst. The feature selection methods produce comparable classifiers, except for the information gain method, which produces worse classifiers for the features evolved with the IG and k-means fitness functions. This is likely because these fitness functions produce many similar highly effective features.

Which fitness function is best is almost certainly problem specific. More study is needed, but from this study, one can conclude that the rF fitness function is best for this problem; that the IG and k-means fitness functions are cheapest to compute and so are best when many mating events are needed; and that the knn fitness function is best when the priority is finding diverse optima of comparable quality. The knn fitness function has parameters: number of neighbours, k, portion of training data used, how often training data and neighbour set are changed. These parameters could affect which features are found.

4.3.6 General Utility Of Evolved Features

String kernels have the nice property that they work pretty well for almost any DNA sequence classification problem. SEM features are more specialized. This is an advantage because it means that a smaller number of features can be used and that analysis of the features can lead to biological insight. It is also a disadvantage because it means that you have to re-run the genetic algorithm for every problem. Therefore, the question of how useful the features evolved to solve one problem were for solving the other, somewhat related, problems was asked. Figures 4.20-4.22 show the distribution of information gain of each set of evolved features together with the distribution of information gain for k-mer features for each problem.

For the sLTR/SINE problem (Figure 4.20) the features evolved to solve the other two problems have similar information gain to the k-mer features. Their classifier performance was also similar with a test error of 1% for a classifier using all the IES features and 0% for a classifier using all the RT features. Their features are less diverse than the string kernel features with a median correlation of 0.3 for the IES features and 0.4 for the RT features, while the string kernel features have a median correlation of 0.1.

The features evolved to solve the other two problems are somewhat more informative than the string kernel features for the IES problem (Figure 4.21). They also do not suffer from the overfitting problem that the classifier made using all the string kernel features

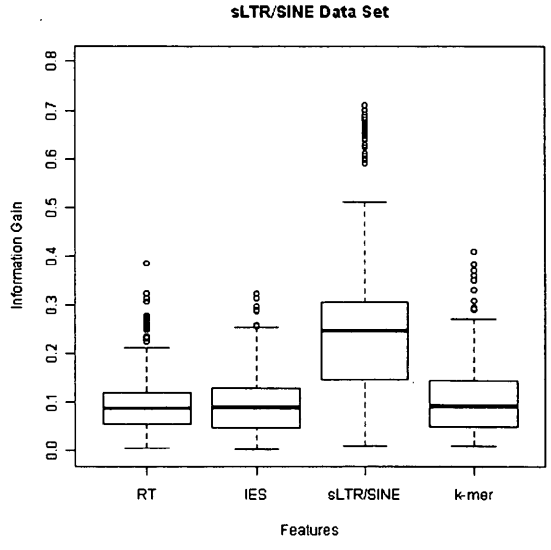


Figure 4.20: Distributions of information gain for the sLTR/SINE data set using string kernel features and features evolved to solve other problems (reprinted from [18]).

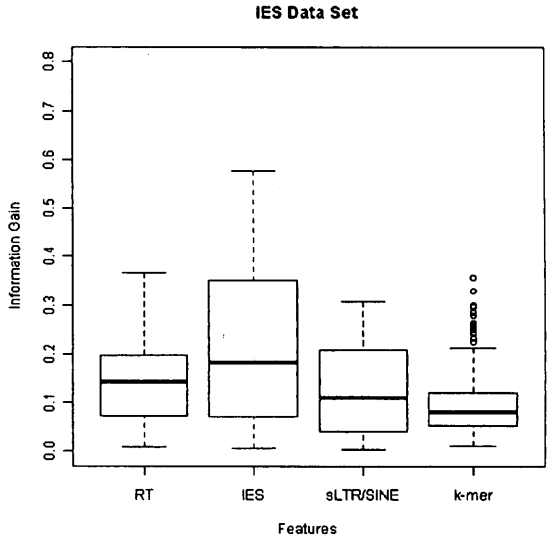


Figure 4.21: Distributions of information gain for the IES data set using string kernel features and features evolved to solve other problems (reprinted from [18]).

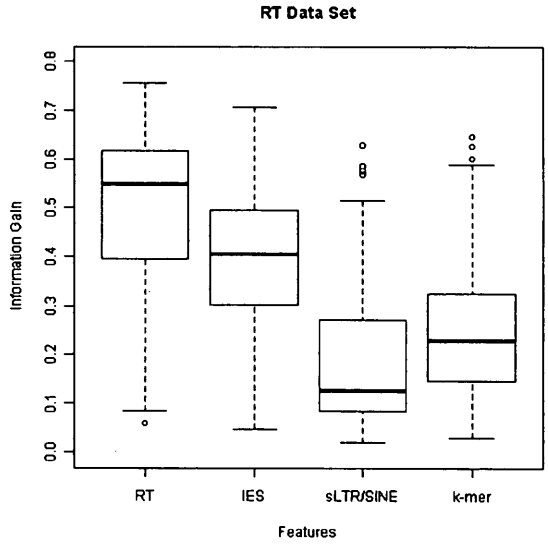


Figure 4.22: Distributions of information gain for the RT data set using string kernel features and features evolved to solve other problems (reprinted from [18]).

had. A classifier created with all the RT features had a 10% test error, only slightly worse than the 8% test error that the classifier made with the features evolved for the problem had. A classifier created with all the sLTR/SINE features had a 16% test error. It is unsurprising that these features were less effective, as they evolved to capture the modular character of the sLTRs.

Figure 4.22 shows the information gain distributions for the features applied to the RT problem. All feature sets have a few really good features. In particular, the features evolved for the IES problem are almost as informative as those evolved specifically to solve this problem. Recall that in Figure 4.17 the IES and RT features appeared to group together when visualized using mutation distance. This result is further evidence that IESs and retrotransposons have similar sequence features, supporting the theory that IESs are mutated transposons. Classifiers built from these features perform comparably to those built from features specifically evolved for the purpose,which have a 6% test error. The classifier built with IES features has 6% test error; the classifier built using sLTR/SINE features has 5% test error. The sLTR/SINE features work better on this problem than they did on the IES problem, because the RTs in this data set contain sLTRs, while the IESs do not.

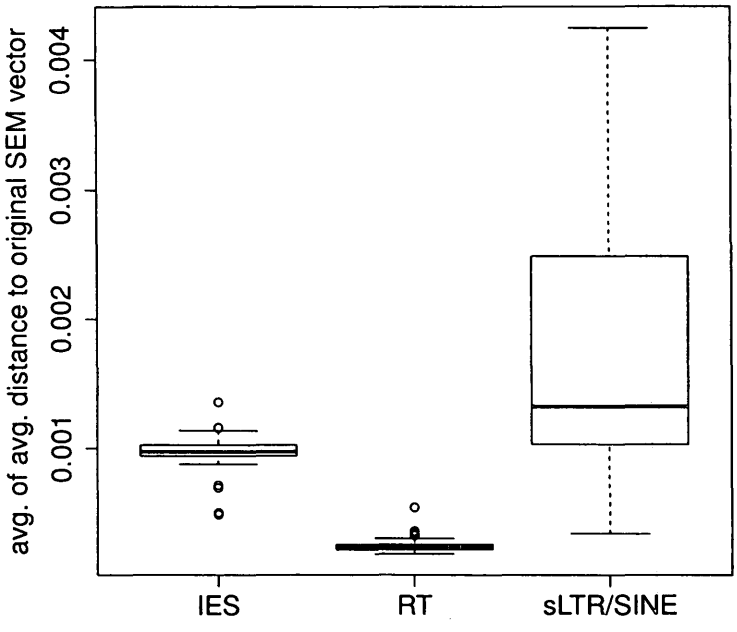


Figure 4.23: Distributions of averages over all sequences in data set of average distances between SEM vector created using original data and SEM vectors created using sequences with an indel mutation. The averages are computed from 100 different indel mutations for each sequence in the data set. Then the average of all the sequence averages is computed. Distributions are over the 100 machines evolved for each problem.

4.3.7 Robustness Of Features To Indel Mutations

Insertion and deletion mutations (indels) are common in biological sequences. Some SEMs can be sensitive to such mutations, such as SEMs that have multiple communicating classes or transient states. A misplaced insertion or deletion in these SEMs could cause a transition that resulted in quite different feature values. Other SEMs are robust to such mutations, making no changes or only small changes to the counts of only a few states. The SEM in Figure 4.2 is an example of this. This is the SEM that calculates purine (A or G bases) and pyrimindine (C or T bases) content of a sequence. Adding or deleting a base would make a small change in one of the counts that would result in an insignificant change in the SEM features for a long sequence.

The impact of indel mutations on SEM features depends not only on the SEMs but also on the character of the sequences on which features are being calculated. A sequence with many repetitive elements, for example, does not change its character much after an insertion or a deletion. A sequence that is best characterized based on a particular motif at a particular position, on the other hand, could become unrecognizable after an unlucky indel. Thus, it is possible to learn about both the behaviour of the SEMs and the character of the classification problem by analyzing the robustness of the SEM features to indel mutation.

Figure 4.23 compares the robustness of the SEM features evolved for the three prob-

lems studied to indel mutation. For each evolved problem, it shows the distribution over the evolved machines of the expected distance between a vector of SEM features based on a sequence in the data set and a vector of SEM features based on that sequence with an indel mutation. The data displayed in this figure was created by first creating a SEM vector for each sequence; then, creating SEM vectors for 100 sequences that differ from that sequence by an indel mutation; then, calculating the distance between each of those vectors and the vector for the original sequence and taking the average; and finally, averaging over all the sequences in the data set.

Notice that the numbers on the y-axis are small. The machine learning classifiers can easily compensate for changes like these and classify the sequences together with their mutated sequences. It is interesting, however, to note the differences between the SEMs evolved for the different problems in this regard. The SEMs evolved to solve the sLTR/SINE problem have both the greatest variation and the largest distances. This is because of the modular character of the sequences in this data set. The SEMs evolved to solve the RT problem are most robust to indel mutations. This is consistent with what is shown in its phenotypic fitness landscape in Figure 4.14. The evolved SEMs for this problem create features similar to k-mer features. K-mer features are also robust to indel mutation.

4.4 Feature Selection

The original approach to SEMs was to use the product of evolution, an individual n-state SEM, in a classifier with n features. This makes it necessary to evolve SEMs with enough states to create classifiers with the desired level of accuracy but with few enough states to avoid overfitting. In [11] it is claimed that the research suggests that SEMs with 12-30 states will produce classifiers with good results on biological data and that SEMs with 36-48 states will overtrain. Working within this range of number of states has two disadvantages. First, SEMs with 12-30 states are difficult to interpret, so the chance to obtain biological insight based on the features is lost. Secondly, it is not possible to add more features in order to get higher accuracies because of the overtraining ceiling. Furthermore, some tests of these classifiers on new data sets suggested that they did not always generalize well.

The analysis in Section 4.3 demonstrates that the SEM fitness landscape is highly multi-modal. This means that the set of effective SEMs produced by evolution in different replicates is diverse. Also, just because a classifier based on an individual SEM is accurate does not mean that all its features are important. The accuracy of the classifier could be driven by just a few of the features, the others just extra baggage. Therefore, features from different replicates were pooled feature selection was done to select the most effective in order to create classifiers with a tuneable number of comprehensible

features.

Standard methods of feature selection did not work well with SEM features. It was hypothesized that this was due to correlation of the features and two alternative feature selection methods chosen to minimize correlation between the selected features were used. The first, called *dissimilarity selection*, is a method used by chemists to select a subset of compounds that is both representative and diverse. The second, called *dissimilarity clustering*, is a novel technique. For comparison, feature selection using established methods is done: the importance measure for random forests and information gain.

Even when there is a multimodal fitness landscape such as exists here, there is no guarantee that different replicates of a genetic algorithm will find different optima. It is possible that every replicate will find the same solution simply because it is the easiest to find. To determine whether diverse features were being found, a correlation matrix was built. Examination of this matrix shows that most pairs of features are correlated, but not strongly correlated. Let $r_{i,j}$ be the Pearson correlation of features i and j. Using consensus sequence data, 99.3% of the pairs of features are correlated (95% confidence using a t-test) with $|r_{i,j}|$ ranging from 0 to 1 with a mean of 0.24. Figure 4.24 shows the distribution of correlations. Notice that low correlations are more common than high correlations. Define *highly correlated* to be $|r_{i,j}| > 0.7$. Then, 5.61% of the pairs are highly correlated. This demonstrates that the features are diverse, albeit correlated. If

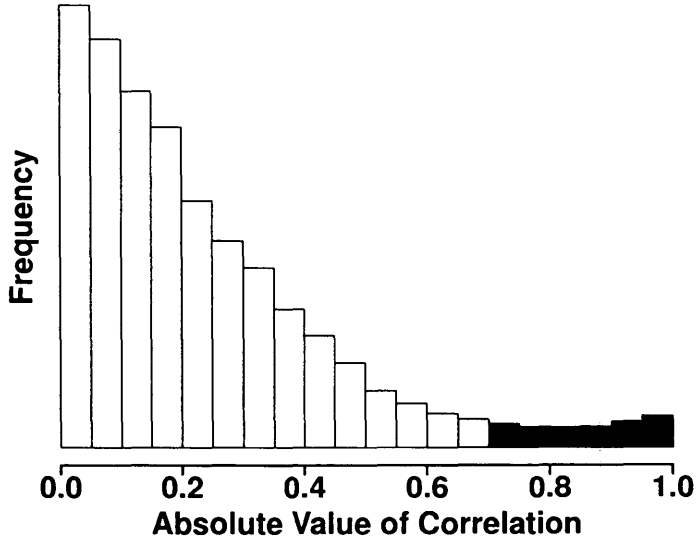


Figure 4.24: Histogram of absolute values of correlations of pairs of features. Filled bars represent highly correlated pairs (reprinted from [17]).

a diverse set of features had not been found, a technique such as niching [88] could have been used to ensure diversity. The existence of a set of diverse quality features is unsurprising because of the large size of the search space and its multimodal character.

4.4.1 Feature Selection Methods In Bioinformatics

Feature selection is an important topic in bioinformatics due to the large number of possible features for many bioinformatics tasks. Feature selection is important to avoid overfitting and eliminate noise, to make classifiers more efficient, and to provide better understanding of the data. There are three basic categories of feature selection techniques: filter methods, wrapper methods, and embedded methods. Filter methods rate

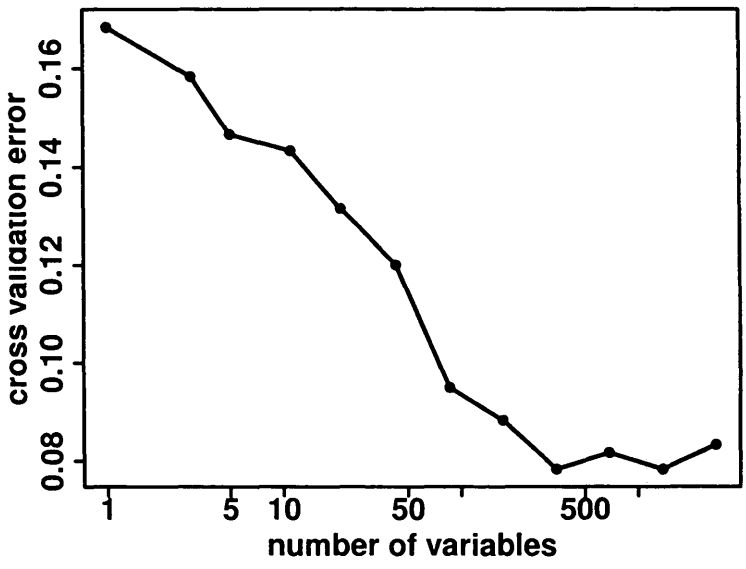


Figure 4.25: Classification accuracy predicted by *rfcv* using different numbers of variables calculated using a data set combining RB, RM, and RT data (reprinted from [17]).

each feature according to some standard, and then choose the best ones. Wrapper methods test the classifier using possible subsets of features. Since the number of subsets is exponential in the number of features, it is necessary to have some method of choosing viable feature subsets. Embedded methods use some property of the classifier to rank features, such as the weight vector in support vector machines. A good review of feature selection methods in bioinformatics can be found in [114].

Important things to consider when choosing a feature selection method include: the computational complexity of the method, whether it takes into account the interactions between the features, whether it takes into account the interactions between the features and the classification method, whether it produces overfitted models, and whether it gen-

erates features which give insight into the data. In general, filter methods have the lowest computational complexity and wrapper methods have the most, with embedded methods intermediate. Filter methods do not take into account the interaction between the features and the classification method and sacrifice their computational simplicity if they take into account the interactions between features. Wrapper and embedded methods take into account the interactions between the features and the classification method, and wrapper methods take into account the interactions between the features. In this thesis, a wrapper method is presented with low computational complexity and compared to an embedded method. Testing for overfitting is done with the use of data sets generated by different methods.

The goal was to use the features to train a random forest classifier. A random forest classifier was chosen because of its ability to avoid overfitting and its insensitivity to noise. Other types of classifiers, such as SVM or AdaBoost, would also work, and the same feature selection techniques could be applied to their use. A common embedded method of feature selection for random forests is to rank the variables according to importance to the classifier and choose a subset containing the most important variables. A technique for determining the correct number of variables to choose was developed in [129]. The function *rfcv* in the R *randomForest* package implements this technique. Figure 4.25 shows the classification error using various numbers of features with this method of feature selection. The figure suggests that the lowest error, 8%, would be

achieved using $\frac{1}{8}$ of the data or 343 features. However, experimentation showed that, in fact, carefully chosen sets with 4-46 variables actually had lower error (data in Section 6.3.5). Therefore, a different technique was needed.

The failure of the *rfcv* technique is likely due to the fact that most of the features are correlated. When several correlated features are used in the same classifier, their importance is underestimated. The idea, therefore, was to create feature subsets containing features as uncorrelated with each other as possible. Two techniques were used to do this, both involving hierarchical clustering: dissimilarity selection (DS) and dissimilarity clustering (DC). *Dissimilarity Selection* (Algorithm 4) involves creating clusters of correlated (or anti-correlated) features and then selecting one feature from each cluster. *Dissimilarity Clustering* (Algorithm 5) involves creating clusters of features with low correlation.

4.4.2 Dissimilary Selection

The algorithm for Dissimilarity Selection is Algorithm 4. Dissimilary selection uses hierarchical clustering. Hierarchical clustering works by partitioning the data in a series of steps which create clusters varying from each object in its own cluster to all objects in a single cluster. These can be represented by a tree with each single object cluster represented by a leaf, and the nodes at each level of the tree representing different possible clusterings. There are different types of hierarchical clustering which do this partitioning

Algorithm 4: Dissimilarity Selection

Data: Dissimilarity matrix D, number of desired features n, selection method, dataset d

Result: Subset *feat* of n features

Create n clusters, $K_1, \ldots, K_n$ using Ward's method with D;
For each K_i, create a submatrix E_i of D;
$feat \leftarrow \{\}$;
for $i \leftarrow 1$ **to** n **do**
    if *method* = random **then**
        $feat \leftarrow feat \cup k \in K_i$ chosen at random
    end
    else if *method* = center **then**
        $feat \leftarrow feat \cup k \in K_i$ whose column in E_i is closest to the mean of the columns of E_i
    end
    else if *method* = best **then**
        $\forall k \in K_i, C_k \leftarrow$ random forest classifier using feature k on dataset d;
        $feat \leftarrow feat \cup k \in K_i$ with smallest error(C_k)
    end
end
return *feat*

in various ways. They all work from a matrix with numerical values relating each pair of objects. The *hclust* function in R [108] with Ward's method was used.

A visualization of the data in two dimensions using multi-dimensional scaling (Figure 6.3) suggested that it does not naturally separate into clear, well-separated clusters. There are features that group together, but without separation between them. Clustering with single-link clustering confirmed this. Single-link clustering is designed to find separations in the data, and, using the data looking for n clusters, it created $n - 1$ singleton clusters and one cluster containing the remainder of the data. This means it was unable to find separations. This is why Ward's method [92] was chosen. Ward's method is biased towards finding clusters of approximately equal size, and it minimizes the variance between members of a cluster. These are both desirable properties here. The object is not to find "true" clusters (as they do not exist), just to partition the data into similar groups of roughly equal size. A different clustering method with these properties (for example, complete link clustering) would not produce identical clusters to the ones used here, though, because are some features that group together, they would not be altogether dissimilar either. It is left for future work to compare the efficacy of applying the techniques used here with different clustering methods.

Usually the matrix used for hierarchical clustering is a distance matrix or a dissimilarity matrix in which smaller numbers represent more similar objects. This means that the clusters formed consist of similar objects. Such a matrix was used for DS. The dis-

similarity matrix used the absolute correlation distance, $1 - |r_{i,j}|$ calculated using the values of the features in the RM data set. Thus, features related by 0 are either perfectly correlated or perfectly anticorrelated. n clusters ($n = 20$ or 50) were created, and an exemplar was chosen from each cluster to create a diverse and representative feature set. Three methods were used to chose the exemplar: *random, center,* and *best.* The *random* method is to select one member of each cluster at random. The *center* method chooses the center by averaging the values in each row of the matrix and then selecting the row most similar to the average. The *best* method is to choose based on classifiers built using individual features, choosing the one with the highest accuracy on a given test set. All three types of data set were used to choose the "best."

4.4.3 Dissimilarity Clustering

Algorithm 5: Dissimilarity Clustering

Data: Similarity matrix S, number of features n, approx. number of features desired m, training data set *train*, testing data set *test*
Result: Subset of features
$k \leftarrow \lfloor \frac{n}{m} \rfloor$ //number of clusters;
//use a *similarity* matrix rather than the usual dissimilarity matrix;
Create k clusters $K_1, \ldots, K_k$ using Ward's method with S;
for $i \leftarrow 1$ **to** k **do**
  $C_i \leftarrow$ random forest classifier using K_i on data set *train*;
  test C_i on data set *test*;
end
return K_i *such that error(C_i) is minimal*

For DC, a similarity matrix was used instead of a dissimilarity matrix for clustering.

This means smaller numbers represent more dissimilar objects. Thus, the clusters consist of objects dissimilar to each other. Each cluster is an instance of the desired dissimilar set. The similarity matrices consist of $|r_{i,j}|$ for each pair of features i and j. These values range from 0 to 1, with features related by 0 being completely uncorrelated, i.e., maximally dissimilar. 137 clusters were created with an average of 20 features each.

4.5 Conclusion

This chapter described how effective SEM features can be selected for identifying and classifying DNA sequences, first by using a genetic algorithm and then by using feature selection on the features from the best SEMs of many replicates of the genetic algorithm. It also examined the fitness landscapes explored by the genetic algorithm using various fitness function and compared the SEM features to string kernel features. The next chapter will show how these selected SEM features can be used for Knowledge Discovery. In Chapter 6 they will be used together with the features from Chapter 3 to solve various classification problems involving TEs.

5 Knowledge Discovery With SEMs

A great advantage of using SEM features for classification is the potential for Knowledge Discovery. Hitherto unknown features of the sequences in question can be discovered by selecting effective features and then analyzing them. Analyzing SEMs, however, is challenging. Finite state machines, especially ones with more than a handful of states, do not convey an intuitive understanding. This is addressed, in part, through developing a method for building effective classifiers with SEMs with a small number of states (Chapter 4). But even with a small number of states, it is necessary to develop further analysis techniques. The first approach used is to relate the SEM features to the more intuitive k-mer features. This approach is tested on the three problems discussed in Chapter 4. Then, more extensive analysis is done on the sLTR/SINE problem.

5.1 Comparison With The String Kernel

For comparison with SEM features, effective k-mer features, $k = 1 \ldots 4$, are examined for the three problems described in Chapter 4: sLTR/SINE, RT, and IES. The features are selected using three different feature selection techniques: random forest importance

Table 5.1: K-mer features selected by different feature selection methods

problem	k-mers	accuracy
rF for RT	T, TT, TTT, TTTT, CTTT, ATTT,	
	TTTC, TTA, TATT, TTAT	90%
IG for RT	T, TT, TTT, TTTT, CTTT, ATTT,	
	TTTC, TTA, GTTT, ATT	89%
DC for RT	A, ACA, AGGA, CCTA, TGTA,	
	TGGC, GTAG,GAAT, GTAT,	
	ATCT, TTCT, AAGT, CAGT, ATTT	69%
rF for IES	AATT, GCT, AAA, AAAA, TTT,	
	TTTT, AGCT, GT, AC, GCTT	88%
IG for IES	AATT, GCT, AAA, AAAA, TTT,	
	TTTT, AGCT, GT, GC, AAAT	88%
DC for IES	TAG, GCT, ACAA, CGAA, CACC,	
	TGGG, CATG, AAAT	68%
rF for sLTR/SINE	CAGG, GGC, GC, AGGC, TTCC,	
	CCTT, CCCT, TTT, GCC, GCCT	97%
IG for sLTR/SINE	CAGG, GGC, GC, AGGC, TTCC,	
	CCTT, CCCT, TTAA, GGCT, GGCG	96%
DC for sLTR/SINE	A, TAC, ATG, CGAA, TAGA,	
	AAAC, CCCC, CTTC, ACCG,	
	AGCG, ACTG, TGTG, ACCT,	
	AGCT, GTTT	94%

Table 5.2: Accuracy of classifiers on test data for random forests trained using various subsets of SEM features.

problem	rF	IG	DC
RT	93%	93%	94%
IES	92%	92%	91%
sLTR/SINE	96%	96%	98%

(rF), information gain (IG), and the novel technique, dissimilarity clustering (DC). Table 5.1 shows the features along with the accuracies of the classifiers created with them. Notice that the ten features selected as most important by the R *randomForest* function seem to follow a common theme and that the highest IG set is almost identical. For the RT problem repeated Ts seem to be a good distinguishing characteristic; for the IES problem both repeated As and repeated Ts are important. DC creates a more diverse set and also includes a larger proportion of the more complex 4-mers. However, the accuracies of the DC classifiers are lower. Having a diverse set of k-mer features does not create effective classifiers since many k-mer features are not appropriate for the tasks.

SEM features are evolved to be effective for the specific problems, but are not necessarily diverse. Table 5.2 shows the accuracies of classifiers created using the three different feature selection methods. The table shows that all feature selection methods produce classifiers with similar accuracies. The rF set and the IG set are more different than for the k-mer features. For the RT problem, 4 out of 10 are the same; for the IES problem, 5 out of 10; and for the sLTR/SINE problem 7 out of 10. This is evidence that, especially for the RT and IES problems, the best features (highest information gain) are also similar features. The similar accuracies for the DC classifiers suggest that using diverse features does not detract from performance.

One way to automatically interpret SEMs is by the nearest k-mer feature and the absolute correlation distance to that feature. Table 5.3 shows this information for the

SEMs selected by DC listed in decreasing order of distance. For the RT problem, all of the SEM features are within a distance of 0.21 of a k-mer. This makes sense, because the k-mer features are well distributed among the evolved SEMs (see Figure 4.14). Note that three k-mers (TG, TTA, and C) have two SEMs that are closest to them. This suggests that the nuances introduced by using SEMs are useful – it is not just C-content that distinguishes the sequence, it is a particular type of C-content.

For the solitary LTR/SINE problem, none of the features are close to k-mers. Even though k-mer features are useful for this problem, evolution found different solutions. This is because of the modular structure of solitary LTRs and SINEs. All of the SEM features come from machines with multiple communicating classes. This means that SEM features are giving a different sort of biological insight into the sequences than k-mer features do.

For the IES problem, about half the SEM features are close (distance less than 0.20) to k-mer features. For this problem, there are even more with the same closest k-mer, with seven close to the 1-mer T. Each of these SEM features are nuanced versions of the T-content of the sequence. Figure 5.1 shows the SEM that generates one of these using State 1. States 1 and 3 resemble the 2-state SEM in Figure 4.2 that computes purine (C or T) and pyrimidine (A or G) content. However, when the sequence is in State 3 and the next base is a purine, it does not return directly to State 1. Instead it goes to State 0 or State 2. Since this feature has a correlation of 0.92 with the 1-mer T, most of the

Table 5.3: Closest k-mers and distance to them for SEMs selected by DC

RT		IES				sLTR/SINE	
k-mer	distance	k-mer	distance	k-mer	distance		
TG	0.21	AAT	0.68	TT	0.13	TAAG	0.79
TG	0.18	AC	0.48	A	0.11	TAAG	0.76
GAG	0.13	AAAA	0.44	T	0.09	AAAT	0.64
T	0.10	AAA	0.42	T	0.08	GAG	0.60
GA	0.09	AAAA	0.37	TT	0.07	TAAA	0.59
ATT	0.08	AAAA	0.37	T	0.06	GT	0.48
TTA	0.08	AAA	0.36	T	0.06	GCC	0.43
TC	0.07	AAAA	0.34	T	0.06	TTT	0.32
TTA	0.07	GT	0.25	T	0.06	TTT	0.32
TA	0.06	GT	0.20	T	0.03	TTTT	0.31
ATTT	0.05	AG	0.18			TTTT	0.24
C	0.04	A	0.18				
C	0.02	AA	0.13				

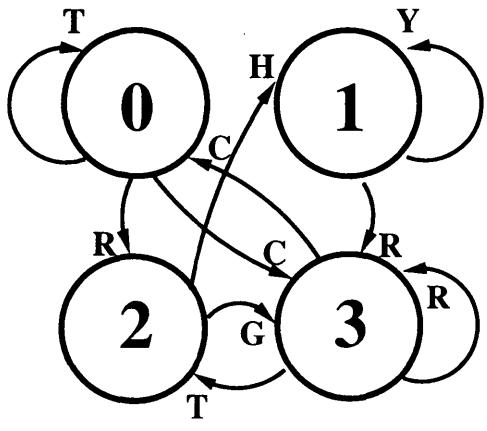


Figure 5.1: The 4-state SEM that generates, using State 1, the IES feature selected by DC that is a distance of 0.08 from the 1-mer T (reprinted from [18]).

Cs in these sequences must occur when the sequences are in States 0 or 3 and most of the Ts must occur when they are in State 1 or 2. Examination of the sequence for the exceptional Cs and Ts could lead to insight into differences between the two types of sequence.

Figure 5.2 shows another SEM that generates a feature closest to the 1-mer T. Of the seven features closest to the 1-mer T, this is the one with the highest information gain, 0.15. The 1-mer T has an information gain of 0.08. State 0 in this SEM counts the T-content of the sequence except when the sequence comes from State 2. In that case, it counts Cs and Gs rather than Ts. Examination of this subset of T-content could lead to biological insight.

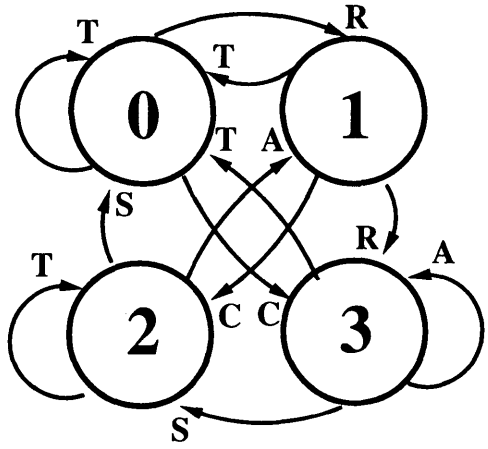


Figure 5.2: The 4-state SEM that generates, using State 0, an IES feature selected by DC that has the highest information gain of the features closest to the 1-mer T (reprinted from [18]).

5.2 Detailed Analysis Of The sLTR/SINE Problem

The sequences studied are solitary LTRs and SINEs. Both are common in the human genome: endogenous retroviral sequences (which include solitary LTRs) comprise 8-10% of the genome [144], and SINEs comprise about 11% [104]. SINEs and solitary LTRs are of similar length and have features in common, and, thus, are easily confused. We know they are functional because they are conserved and transcribed [60, 102, 95], but their function is not well understood. SINEs are related to small RNAs, while solitary LTRs are related to promoter regions for genes. Because they insert copies of themselves in multiple locations in the genome, both solitary LTRs and SINEs impact genome size and structure, but differently due to different insertion site preferences. Identifying families of solitary LTRs and SINEs is important to the the study of genome evolution as they serve as biological markers. See Section 1.2.4 for more information about solitary LTRs

and Section 1.2.5 for more information about SINEs.

Genomic data is constantly being updated and revised. There are often conflicting annotations for genomic structures, and annotations are frequently modified. For example, the RepeatMasker and RetroTector annotations extracted for solitary LTRs are mostly disjoint. On human chromosome 19 only 20% of the annotations overlap. The impact these different annotations had on the results for the sLTR/SINE problem was studied. Three different data sets were used for the solitary LTRs: the one used in Chapter 4 and Section 5.1 that uses RetroTector's annotation of LTRs from their identified LTR retrotransposon sequences (RT data set); complete solitary LTRs identified by RepeatMasker (RM data set); and solitary LTRs catalogued in RepBase (RB data set). The RT and RM data set are comprised of sequences taken directly from genomes. They are sometimes referred to as genomic data sets. The RB data set consists of consensus sequences derived from many instances of the same solitary LTR. This data set is sometimes referred to as the consensus data set.

All consensus sequences in RepBase designated as human solitary LTRs or human SINEs are used. The sequences from RepeatMasker are taken from human chromosomes 1 and 2 and include sequences annotated by RepeatMasker as being complete matches for solitary LTRs and SINEs (i.e., having no gaps at the beginning or end of the match). The sequences from Retrotector are taken from annotations of HERVs in the output from a search using their online tool on human chromosomes 19 and 21. Since Retrotector does

Table 5.4: Experiment Sets – M is the machine number; s is the state number.

Method	States	Reference
non-looping	4	4s$M(s)$
non-looping	6	6s$M(s)$
looping	4	4sL$M(s)$
looping	6	6sL$M(s)$

not generate any annotations for SINEs, the Retrotector data sets combine Retrotector solitary LTRs with RepeatMasker SINEs. The RM and RT data sets are divided into training and test data sets.

5.2.1 Experiments

Four sets of experiments were performed with 100 replicates each. 4- and 6-state SEMs were used. For each number of states, two different methods of running the sequence through the SEM were used. The sequence always starts in State 0. For the first method (non-looping), the sequence is run through the machine from beginning to end. Since the sequences vary in length, it was hypothesized that this might create uneven results. So, the second method (looping) normalizes by always using 10,000 steps, looping from end to beginning of the sequence. For each replicate, the machine with best fitness is saved for analysis. The set of 100 best SEMs, one from each replicate, are used to generate either 400 or 600 (depending on whether 4 or 6 states are used) features. Subsets of these features are used to train random forest classifiers (see Section RFsection). One hundred replicates were done to maximize the diversity of the chosen features. The SEM

search space is large, and features taken from the same SEM are necessarily correlated. Using 100 replicates ensures that there will be subsets of features with small correlations. In addition, 100 non-looping replicates were done with 20-state machines in order to provide a comparison of the new method with the old method of creating a classifier based on a single machine.

A somewhat arbitrary choice was made to use the RB data for the genetic algorithm. The rationale was that this would mean the features would be based on underlying features rather than mutations. The results, which demonstrate that features based on mutations can be important identifiers, suggest that it would be worthwhile to try evolving with the genomic data sets in the future.

5.2.2 Non-SEM Features

In addition to the features generated by the SEMs, the following features were included. These features are included because they have been shown to be useful for similar classification problems. Their use provides help in interpreting SEM features that are similar to them and serves as a basis of comparison for measuring the effectiveness of SEM features. More details about these features can be found in Chapter 3.

- The length of the string (Section 3.3).
- Dinucleotide frequencies (2-mers) (Section 2.1.4).

- Shannon entropy of 2-mers, 3-mers, and 6-mers (Section 2.1.5).
- Purine (A or G) content, amino base (A or C) content, strong H-bond base (G or C) content (Section 3.3).
- Gap features (Section 3.3).
- Maximum length of runs for each base (Section 3.3).

5.2.3 Impact Of Source Of Data

Many researchers who use machine learning approaches to genomic analysis use consensus sequence data for training, for example TEclass [2] and REPCLASS [47]. Preliminary investigations of the features, however, indicated that training with the consensus sequences from RepBase was not optimal. A striking demonstration comes from examination of the entropy features. For the RB data set, these features appear to be excellent. Using just the three entropy features, the OOB training accuracy is 97%. However, the accuracy of this classifier on the solitary LTR sequences in the RT data set is little better than random guessing, 54%.

Figure 5.3 shows a projection of the solitary LTR data using four of the features (the selection of which is described in Section 6.3.5) into two dimensions using multidimensional scaling. Notice that the consensus sequence solitary LTRs, shown as circles, group together in a line, while the genomic solitary LTRs, shown as triangles, pointing

Table 5.5: Accuracy of classifiers distinguishing sequences found by RepeatMasker from consensus sequences.

features	sLTR acc.	SINE acc.
20 cluster centers (Section 6.3.4.1)	73%	95%
entropy features	93%	98%
minGC..CG, CG-content, CG freq.	95%	97%
CG-content, freq. of AT, CG, GC, GG, and TA	99%	99%

up for RM and down for RT data, are more scattered. Although a visualization like this does not prove anything, it suggests that the problem with using consensus sequences for training is that their feature values have much tighter distributions than those of genomic sequences.

To further investigate this phenomenon, classifiers were built to distinguish consensus sequences from genomic sequences. If consensus sequences are truly representative of genomic sequences, these classifiers should have low accuracy. In fact, they did not. Some of the features in the feature set can distinguish a consensus sequence from a genomic sequence with high accuracy. See Table 5.5 for accuracy of classifiers built from these features. The fact that such classifiers exist demonstrates that consensus sequences are not always representative of genomic sequences. The results suggest that consensus sequences should be used with caution, and that a comparison should be made with classifiers built from genomic sequences.

One possible explanation for the difference between consensus and genomic sequences involves the mechanism of mutation. Consensus sequences are built by aligning

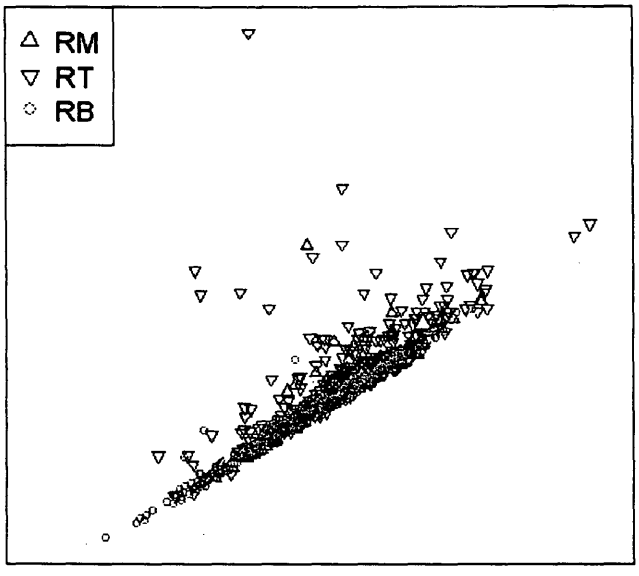


Figure 5.3: Projection into two dimensions of solitary LTRs from the three different types of data represented using the four super-features (reprinted from [17]).

a set of genomic sequences and taking the majority vote at each location. The assumption is that mutations occur in random locations. This means that the process of building a consensus sequence filters them out. The original base, not the mutation, will always be in the majority in a particular position. However, the mutations are not in fact random. For example, it is known that G-to-A mutations targeting GA and GG dinucleotides are common in many human retroviruses and retroelements [8]. These would affect CG-content and the frequency of GG dinucleotides, two features that were used to distinguish consensus from genomic sequences. As a result of this analysis it was decided to build classifiers using all three training data sets and to compare the results.

5.2.4 Dissimilarity Selection

The accuracy of classifiers built for this problem using DS will be discussed in Section 6.3.4.1. In this section, it is shown how DS can be used to better understand the SEM features. Features that cluster together are likely to play a similar role in classifiers. Understanding each cluster provides understanding of the range of SEM features found by the genetic algorithm. Performing DS on evolved SEM features together with statistical features has a dual purpose: it pinpoints potentially valuable features that the genetic algorithm fails to find, and it helps with the analysis of SEM features that cluster together with the more easily interpreted statistical features.

Table 5.6: Twenty clusters used for DS. For mixed types, the number in parentheses is the percentage of features that are evolved. Highlighted items are discussed in the text.

name	size	med. acc.	best acc.	cluster acc. RM	cluster acc. RB	type
Starting Pyrimidines	169	90%	**92%**	92%	91%	SEM looping
Starting Seq. A	221	72%	**96%**	94%	80%	SEM non-looping
Starting Seq. B	140	71%	**92%**	94%	79%	SEM non-looping
Starting Seq. C	111	69%	**92%**	93%	79%	SEM non-looping
GG Max Gap	89	62%	79%	82%	78%	non-evolved
Length	108	62%	73%	79%	76%	non-evolved
Entropy	154	59%	69%	74%	75%	SEM non-looping
Assorted	247	58%	**96%**	84%	74%	mixed (38%)
A-rich	176	56%	75%	**88%**	74%	SEM (all types)
Max Gap Duds	60	56%	68%	79%	74%	non-evolved
Sequence Comp.	214	54%	69%	**93%**	73%	mixed (81%)
GG Cluster	57	54%	75%	**89%**	72%	non-evolved
CT/TC Cluster	56	54%	63%	**88%**	70%	mixed (50%)
GC-Content	309	53%	65%	**93%**	**69%**	mixed (90%)
Mutation Cluster	76	53%	**93%**	92%	**60%**	SEM looping
CC Cluster	55	52%	71%	**90%**	**60%**	non-evolved
GC Cluster	99	52%	65%	**91%**	**58%**	mixed (74%)
Mutated A-rich	160	52%	76%	85%	**53%**	mixed (86%)
Amino Cluster	164	52%	66%	**88%**	**51%**	mixed (73%)
CG Cluster	77	51%	63%	76%	**42%**	non-evolved

5.2.4.1 Analysis Of Clusters

Table 5.6 lists some basic statistics about the clusters. The clusters are named so as to give some indication of the nature of the features they contain. They are listed in order from best to worst in terms of their accuracy when all their features are used in a classifier trained on RB data and tested on RT data (the hardest combination of training/test data).

The clusters range in size from 50 to 309 features. Since there were more than twice as many evolved as non-evolved features, the clusters with non-evolved features tend to

be smaller. Large clusters with evolved features include similar features found by many replicates of the genetic algorithm.

About a third of the clusters consist of 98% or more evolved features, a third of 98% or more non-evolved features, and a third are mixed. This means that the genetic algorithm is finding features similar to some of the non-evolved features (those in the mixed clusters), features different from any of the non-evolved features (those in the evolved clusters), and failing to find features similar to some of non-evolved features (those in the non-evolved clusters). In addition, the different types of SEM experiments (looping and non-looping, 4-state and 6-state) distribute their features in different clusters. This demonstrates that doing the different experiments led to a greater range of features and suggests that new experiments with different parameters could find different effective features.

Many of the sequences in the problems being studied have a modular sructure. Most of the evolved SEMs had one transient and one attracting communicating class. This enabled them to divide the sequence into two modules: the initial portion, and the rest.

5.2.4.2 Effective Individual Features

The median accuracy column in Table 5.6 demonstrates that most features create weak classifiers by themselves. Examination of the best accuracy column shows that exceptional features occur in six clusters: the four clusters that measure qualities of the starting

portion of the sequence (Starting Pyrimidines Cluster and Starting Sequences Clusters A, B, and C), the Assorted Cluster, and the Mutation Cluster. These features create classifiers with better than 90% accuracy. They are shown in bold. All of these are evolved features. These six clusters have the property that better classification can be achieved by using the single best feature than by using a combination of their features.

The four top clusters in Table 5.6 all measure qualities of the starting portion of the sequence. The large Starting Pyrimidines Cluster contains highly effective features very similar to each other. It essentially has two features that are found in many evolutionary replicates of looping SEMs. The larger of its feature groups contains features that measure the proportion of the sequence consisting of pyrimidines before the first purine; the other group has features that measure the proportion of the sequence consisting of initial Ts. For both types of features, a solitary LTR with a typical TG start has a small value, and the value for the T-rich 5' region of a SINE is larger. An outlier, distance 0.10 from the larger group, gets slightly higher accuracies. It also measures pyrimidine content in the starting part of the sequence, but it has a more complicated definition of "starting part of the sequence." It considers starting sequences with forms like: R^*YC^*R, R^*YR, and R^*YTR^*YR.

The three clusters called Starting Sequences Clusters A, B, and C also measure qualities of the starting portion of the sequence. They consist of SEM features created by non-looping SEMs that, in general, work better when trained with genomic sequences

than with consensus sequences. Starting Sequence Cluster A contains twelve of the top twenty individual features, all of which are highly correlated with each other. One of these counts the proportion of Ts in starting sequences. It considers starting sequences including: C, RS, TG, TM^*G, TRT^*S, and MT^*AT^*S.

Most features in Starting Sequences Cluster B are highly correlated with the cluster centre, which counts various starting sequences including the typical TG start of solitary LTRs and also sequences such as: CG, $CTCG$, $TTCG$, $CATG$, GA^*CG, and AG^*TA^*CG.

An exceptional feature in Starting Sequences Cluster B is on the edge of the cluster. It is not close to any other feature in the feature set. This feature is 6s77(1) [8]. The same SEM that produces it also produces 6s77(3), a highly effective feature in the Assorted Cluster. This SEM is shown in Figure 4.1. 6s77(1) is unusual in part because it is derived from a transient state that can only be reached from a transient communicating class. This is an unusual SEM structure. Most evolved SEMs divide into a transient and an attracting communicating class without transient states between them. 6s77(3) is non-zero for starting sequences $G(SY)^*W$, $G(SY)^*SR$, TYW, $T(YS)^*R$, and $T(YS)^*YW$ and zero otherwise. Its value is usually zero for solitary LTRs and usually non-zero and larger for SINEs, meaning that it is identifying distinctive starting sequences of SINEs. 6s77(1) counts the number of Cs between these starting sequences and the rest of the

[8]Feature names are of the form $NsM(i)$ where N is the number of states; M is the machine number; i is the state number. Features from the looping experiments replace "s" with "sL".

sequence. This feature is interesting because it is detecting an intermediate module in SINE elements. Its success suggests that it would be worthwhile to explore modifications to the genetic algorithm that would encourage discovery of multiple modules in the sequences.

Starting Sequence Cluster C consists of a small but varied set of starting sequence features. The cluster centre, a highly effective feature, counts, for a sequence that begins with CA, CG, TA, or TG, the number of TAs, TGs, As and Gs until it reaches a C.

The Assorted Cluster was given its name because its features are not highly correlated with other features. This means it contains features infrequently discovered by the genetic algorithm. It is the second largest cluster and mixes evolved and non-evolved features. The best features in this cluster include 12 SEM features and the SINE motif CCTT found by a minimum gap feature. Although many of the non-evolved features contained in it are useful, the more nuanced SEM features similar to them are more useful. Interpretation of the SEM features is aided by their proximity to non-evolved features which include: entropy of 2-mers and 3-mers, maximum size run of Gs, and the average gap between GC and AG.

5.2.4.3 Features Effective In Combination

Some clusters create better classifiers when their features are used in combination, especially when trained using genomic data. Notable examples are shown in bold in the

cluster accuracy (trained on RM) column. The only one of these consisting entirely of evolved features is the A-rich Cluster. It seems to be identifying features of the A-rich modules of solitary LTRs identified by [20]. The CC Cluster and the GG Cluster contain average and minimum gap features involving the dinucleotides CC and GG that were not found by evolution, perhaps because they are only effective in combination. Five other clusters contain a mix of evolved and non-evolved features that are effective only in combination. These are: the Sequence Composition Cluster, the GC-Content Cluster, the GC Cluster, the CT/TC Cluster, and the Amino Cluster. These clusters all contain average gap features, dinucleotide frequency features, and SEM features. Three of the five have primarily 6-state SEM features. This highlights a reason why features from SEMs with fewer states are more interpretable. Not only are the SEMs themselves easier to interpret, but also each individual feature contributes more to the classification.

Examples of SEM features from these clusters:

- A feature in the A-rich Cluster calculates the A-content of the sequence plus Cs and Gs that follow TT or GT and all bases following Cs that do not follow TT or GT.

- A feature in the GC-Content Cluster counts runs of As that follow sequences of the forms: GY^*A, DM^*T, and CT^*C.

- A feature in the GC Cluster counts Ts that follow sequences of the forms: MG,

TW, A^*G, CA, GR, and TC^*W.

- A feature in the CT/TC Cluster counts runs of Cs that follow Ts.
- A feature in the Amino Cluster that is difficult to analyze involves the base content in the final portion of the sequence, paying special attention to the number of Cs.

5.2.4.4 Features That Work Only On Genomic Sequences

The features in some clusters are more sensitive than others to whether they are trained using genomic or consensus sequence data. The clusters that are most sensitive to the type of training data are shown in bold in the cluster accuracy (trained on RB) column. Further study of the features in these clusters could lead to insight about mutation biases. Two of these clusters include only non-evolved features, the CC Cluster and the CG Cluster. These contain features related to the CC and CG dinucleotides. The Mutation Cluster consists entirely of evolved features. Four other clusters have a mix of evolved and non-evolved features: the GC-Content Cluster, the GC Cluster, the Mutated A-rich Cluster, and the Amino Cluster.

The GC-Content Cluster is the largest cluster, meaning that its features are frequently rediscovered in replicates of the GA. It contains all the features in the classifier from Table 5.5 that achieved 99% accuracy distinguishing consensus sequences from genomic sequences. The GC Cluster contains feature related to the GC dinucleotide. Their success suggests that GC dinucleotides occur more frequently and closer together in SINEs than

in solitary LTRs. Like the A-rich Cluster, the Mutated A-rich Cluster is likely detecting features in the A-rich regions of solitary LTRs. However, since it performs better when trained with genomic sequences, it is detecting aspects that occur only after mutation.

5.2.4.5 Features That Genetic Algorithm Does Not Find

While the average and minimum gap features sometimes cluster with evolved features, the maximum gap features never do, suggesting either that they measure qualities difficult to represent using SEMs or that SEMs are able to find better features. Since only 10 out of 256 of these features create individual classifiers with accuracy 75% or above, the latter is probably the case. The maximum gap features are divided amongst four clusters: GG Maximum Gap Cluster (containing maximum gap features that contain the dinucleotide GG), Maximum Gap Duds Cluster (containing the least effective maximum gap features), Length Cluster (features correlated with the length of the sequence), and CG Cluster (features associated with the CG dinucleotide). The most effective of them fall into the GG Maximum Gap Cluster. These all involve the dinucleotide GG and tend to be larger for solitary LTRs, indicating that GGs are more spread out in solitary LTRs than in SINEs. Interestingly, the frequency of GG is in Starting Sequence Cluster C and is an average distance of 0.42 from these features, suggesting that these features are measuring a quality unrelated to it. The Maximum Gap Duds Cluster contains the least effective of the maximum gap features. The Length Cluster contains about half the

maximum gap features along with the length of the sequence. The sequence length is a highly effective feature when trained with genomic sequences, less so with consensus sequences, suggesting that solitary LTRs are more likely than SINEs to have insert mutations. The remainder of the maximum gap features are in the CG Cluster, which contains non-evolved features related to the CG dinucleotide.

5.2.4.6 Randomness

SINEs have lower entropy than solitary LTRs, because they often have short sequence repeats at one or the other end, such as GGCTGGCTGGCT. This reduces their entropy. They also end with poly-A tails, AAAAAAAAAA, that further reduce their entropy. The Entropy Cluster contains features that use this fact to distinguish the sequences. It consists almost entirely of SEM features, but also includes the 6-mer entropy feature. The cluster centre, for example, detects repeats of the subsequence TM^*TS occurring at the beginning of SINEs.

5.2.5 Dissimilarity Clustering

The accuracies of classifiers built using dissimilarity clustering is discussed in Section 6.3.5. DC is valuable for biological analysis because it selects a diverse set of effective features. The features in an effective DC classifier give insight into different distinguishing qualities of the sequences. For this problem, both the best DC classifier trained on

consensus sequence data and the best DC classifier trained on genomic data contained the same four features. These are referred to as the four super-features.

These four super-features are:

1. **4s83(2):** This feature is in the Assorted Cluster. It is on the edge of the cluster, with a correlation of 0.11 with the cluster centre. It is highly correlated only with another feature generated by the same SEM and two pairs of features from similar SEMs, meaning it is rarely found by evolution – only three times out of 400 replicates. It is part of a transient class identifying a sequence start feature. Its value is zero if the sequence starts with TG or A (i.e. for most solitary LTRs). It adds together the lengths of some of the runs of amino bases in the starting sequence. For example, those following an initial C, G, or TT, and those following TTT, but not those following an initial T or those following GT or a non-initial TT. This feature achieves 96% accuracy on RM training data and 95% accuracy on RT data when used alone.

2. **4s63(2):** This feature is a member of the Starting Sequences Cluster C. It is in the interior of the cluster with a correlation of -0.52 with the cluster centre, and, like the cluster centre, measures qualities of the start of the sequence. It is a unique feature, highly correlated only with two other features generated by the same SEM. It is a terminal state that counts the proportion of the sequence following certain

possible starting sequences. If the sequence does not start with T, its value is 1. Most SINEs fall in this category. Starting sequences excluded from its count include: TT, TGS, TGT^*S, $TGAT$, TM^*T, and TM^*GT^*S. Solitary LTRs are more likely to have longer such starting sequences than SINEs. This feature achieves 90% accuracy on RM training data and 93% on RT training data when used alone.

3. **6sL12(3):** This feature is highly correlated with the frequency of TT ($r = 0.95$) as well as 69 other features. It is a member of the Sequence Composition Cluster and has a 0.59 correlation with the cluster centre, putting it in the interior of the cluster. It measures the frequency of runs of Ts together with CCs (i.e., sequences like TTTTTCCTTCCCCTTTT, but not like TTTTCTTTC). About 8% of solitary LTRs consist of such sequences, while the amount in SINEs is variable. This feature achieves 75% accuracy on RM training data and 83% accuracy on RT training data when used alone.

4. **4sL66(1):** This feature is in the interior of the Mutation Cluster with a correlation of 0.43 with the cluster centre. It is nearly unique with high correlation only with features generated by two other 4-state looping SEMs. It has a 0.67 correlation with amino content and counts all amino bases except runs of Cs following a G or T. In addition, it counts Ts that follow a G or a T and Ts that follow KC^*.

This feature achieves 70% accuracy on RM training data and 79% accuracy on RT training data when used alone.

5.3 Conclusion

This chapter compared the impact of feature selection on SEM features with that on string kernel features. It showed that, for SEM features but not for string kernel features, a feature selection method that selects diverse features is effective. It introduced some methods for SEM feature analysis: finding the closest string kernel (or other more easily interpreted) feature, clustering the features, and direct analysis of the finite state machines that generate the SEM features. An important discovery made in the course of the analysis of SEMs selected for the sLTR/SINE problem was that consensus sequences may not always be a good choice for machine learning training sets for DNA sequence classification problems. Dissimilarity clustering was used to find a small set of highly effective diverse features for the sLTR/SINE problem. The next chapter will use the SEM features along with the statistical features from Chapter 3 in various DNA sequence classification problems involving TEs.

6 Classification Problems

This project was started with the goal of building a scanner to detect ERVs. The first step towards building such a scanner is to build classifiers that distinguish ERVs from other genomic features. This includes intergenic non-coding sequences, genes, and other types of TEs. Both classifiers using the statistical features and classifiers using SEM features were built. In addition, the two types of features were combined by creating SEMs that are driven not by the {A,C,G,T} alphabet, but by the pattern of reading frames detected in the sequence. High accuracies were obtained with all of these classifiers.

6.1 Types Of Classifiers Used

Three types of machine learning classifiers were used: SVMs, random forests, and k nearest neighbour. These classifiers perform well with the data. Other types of classifiers might work as well or better. No attempt was made to optimize based on classifier type, comparing only the performance of SVM and k nearest neighbour classifiers in Section 6.4. That comparison is done to make sure that there is no bias towards using the classifier that is part of the genetic algorithm's fitness function. In general, It is likely that only

small improvements could result from optimizing the classifier used since the accuracies are high. Given the noisiness of the data, these improvements would be unlikely to be meaningful.

6.1.1 Support Vector Machines

SVMs are supervised learning algorithms for classification and regression. They are useful for dealing with data that is noisy and/or not linearly separable. They have become popular in many diverse applications where efficient and accurate classifiers are desirable. SVMs learn hyperplanes from training data that separate the classes and maximize the distance (margin) to the nearest training data points on either side of the hyperplane. Since SVMs maximize the geometric margin, they are also known as maximum margin classifiers. Since the data is not linearly separable, the hyperplane must be in a higher dimensional space than the data. This is achieved through use of a kernel function. The kernel function maps the data onto a feature space in which they are linearly separable. The support vectors are vectors taken from the training data set that lie on the margin. It can be shown that it is not necessary to calculate the maximum margin hyperplane itself. The classification can be done using a function of the support vectors. The number of support vectors required by the machine is some indication of the complexity of the model. Although a hyperplane divides space into two classes, SVMs are easily adapted to handle multi-class problems. More details on SVMs can be found in [30]. The LIBSVM

library in R [35] was used to train and test the classifiers.

The choice of SVM for classification was based on several factors. SVMs are commonly used in bioinformatics because they work well in high dimensional spaces. This gives the flexibility being able to use many parameters without worrying about the "curse of dimensionality." The classes are not linearly separable, and the data is noisy. SVMs have a parameter, C, that allows one to compensate for noise. A disadvantage of SVMs is that they are not transparent. For this reason, random forest classifiers were used later, which have the same advantages as SVMs, but are easier to interpret and analyze.

6.1.2 Random Forests

A random forest is an ensemble classifier made of decision trees [26, 54]. The classification is made by majority vote of the trees. Each tree is trained using a different subset of $N\%$ of the data. At each node, a subset of m features is chosen. The best cutoff value for each of the m features is determined, and the feature that splits most equally, measured using the Gini index, is chosen. The trees are not pruned.

The Gini index is a measure of inequality, ranging from zero (completely equal) to one (completely unequal), often applied to income. Canada, for example, has a Gini index for income around 0.32, while the United States has one around 0.47, and humanity as a whole has one around 0.65. The Gini index is measured using the Lorenz curve. To create a Lorenz curve, you sort the feature values, $x_1 \leq x_2 \leq \ldots \leq x_n$, and then plot the

points $(h/n, \sum_{i=1}^{h} x_i / \sum_{i=1}^{n} x_i)$ where $h = 1 \ldots n$ and join the values, together with the point (0,0), with a polygon. Let A be the area between the line of perfect equality (the diagonal) and the Lorenz curve and B be the area under the Lorenz curve. Then the Gini index is

$$\frac{A}{A+B}. \tag{6.1}$$

The random forest yields an *out of the bag* (OOB) classification error, which is the percentage of misclassifications when the remainder of the data, not the data used for training, is classified by each tree. It is common to use between 50 and 500 trees. The random forests are created using code from alglib[9] [25].

An advantage of random forests is that they are unlikely to over fit the data. The inventors of the technique claimed that it could not over fit. Others have disputed this, but, in any case, it is not prone to overfitting. The software package used allows adjustment of a parameter to correct for possible overfitting. Experimenting with this parameter determined that overfitting was not a problem in this case. Random forest classifiers are also a good choice for noisy data. They are less sensitive to noise than other classifiers, since, if one tree fails to identify a sequence due to noisy data for one feature, another tree that relies on a different feature can spot it. Random forests are transparent classifiers. It is possible to analyze the trees and learn something about the sequences.

A useful tool for sequence analysis is the random forest distance created using a

[9] www.alglib.net

high accuracy random forest classifier. Each sequence that passes through a random forest ends up in a specific leaf of each decision tree. Each tree has many leaves that result in the same classification. The random forest distance between two sequences is the percentage of trees in which they end up in the different nodes. It is possible for two sequences with the same classification to have the maximum possible random forest distance. This happens when they follow a different path in each decision tree. It is not possible for sequences with different classifications to have the minimum random forest distance, but they can have a small distance if both sequences are misclassified on many of the trees and their misclassifications follow the same paths as the correct classifications for the other sequence.

6.2 Classifiers Using Statistical Features

Attention was first focused on the problem of detecting ERVs. A difficulty in detecting ERVs is that they are mutated and so may have lost some of their distinctive retroviral features. Thus, the presumably easier problem of distinguishing exogenous (wild) retroviral genomes from other genomic sequences was explored first. Since human ERVs (HERVs) are most commonly found in non-coding DNA, a classifier was built to distinguish exogenous retroviral genomes (RVs) from non-coding human sequences (NCSs). Next, an SVM classifier was built that distinguished HERVs from NCSs. As some HERVs are found in genes, an SVM classifier was built to distinguish HERVs from

human genes. Out of curiosity, it was then investigated whether distinctions could be made between different types of viruses. The algorithm's ability to distinguish HERVs from RVs, lentiviruses from other retroviruses, papilloma viruses from lentiviruses, and papilloma viruses from retroviruses was tested. In all cases, the SVM classifiers achieved high accuracy.

6.2.1 Data Sets

For this first set of classifiers four types of data were used: human endogenous retrovirus (HERV) data, viral genome data, non-coding sequence (NCS) data, and human genes (GENE). The viral genomes are divided into several data sets: retroviruses (RV), which are subdivided into lentiviruses (LENTI) and retroviruses that are not lentiviruses (NON-LENTI), and papilloma viruses (PAP). Lentiviruses are a type of retrovirus (the type that includes HIV), and papilloma viruses are not retroviruses but have some genes in common with them. The HERV data set was created using RetroSearch and the viral genome data was obtained from NCBI[10]. The NCS and GENE data sets contain sequences taken from the human genome downloaded from NCBI. The NCSs were selected at random, excluding regions that are known to be genes or HERVs. The HERV regions that were excluded were taken from RepeatMasker. The GENE data was selected at random from genes mapped in NCBI that are at least 5000 and not more than 10,000 nucleotides long.

[10] `http://www.ncbi.nlm.nih.gov`

They include both exons (coding regions) and introns (non-coding regions). The 356 HERVs in the HERV data set were chosen to have minimum length 5000, minimum open reading frame (ORF) length 100, at least 3 ORFs, and a minimum identity with known retroviruses of 90%. The RV data set has 58 complete retroviral genomes. LENTI, NON-LENTI, and PAP have 96 genomes each. NCS data sets for each experiment were chosen to have the same number of sequences as the other data set in the experiment with the same distribution of lengths. Gene data sets, also, were chosen to have the same number of sequences as their companion data set.

6.2.2 Features

For these experiments, Fourier transform based features derived from the RY, MK, and SW indicator sequences were used. These include Fourier magnitude features (Section 2.1.2) to detect coding regions and Fourier phase vector features (Section 3.1.3) to detect the reading frame pattern. Entropy features (Section 2.1.5) were used to measure the randomness of the sequences. See Table 6.1 for a summary of the features used.

6.2.3 Distinguishing Retroviruses From Non-coding DNA

The problem expected to be easiest was studied first: distinguishing retroviruses (RV data set) from non-coding DNA (NCS data set). The results are shown in Table 6.2. The table shows the average of 200 iterations and best results for sensitivity, specificity and

Table 6.1: Features used for classification

Abbrev.	Description
mRY	mag. of Fourier coeff. $S(\frac{N}{3})$ in RY string
mMK	mag. of Fourier coeff. $S(\frac{N}{3})$ in MK string
mSW	mag. of Fourier coeff. $S(\frac{N}{3})$ in SW string
f(1)	freq. of distance 1 betw. change points
f(2)	freq. of distance 2 betw. change points
f(3:5)	sum of freq. of distances 3, 4, and 5
f(6:9)	sum of freq. of distances 6, 7, 8, and 9
avgblk	avg. distance betw. change points
e1	entropy of single bases
e2	entropy of dimers
e3	entropy of trimers
e6	entropy of hexamers

Table 6.2: Results for RV-NCS classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.98	1.00	1.00	1.00	0.99	1.00	58%
e1, e2, e3, e6	0.99	1.00	0.98	1.00	0.99	1.00	24%
mRY, mMK, mSW	0.82	1.00	0.81	1.00	0.81	0.95	57%
f(1), f(2), f(3:5), f(6:9), and avgblk	0.74	1.00	0.58	0.82	0.66	0.86	58%

accuracy of 200 trials using random selections of the data for training and testing. These are calculated using the following equations:

$$\text{sensitivity} = \frac{tp}{tp + fn}, \tag{6.2}$$

$$\text{specificity} = \frac{tn}{tn + fp}, \tag{6.3}$$

and

$$\text{prediction accuracy} = \frac{tp + tn}{tp + fn + tn + fp} \tag{6.4}$$

where tp = number of true positives, tn = number of true negatives, fp = number of false positives, fn = number of false negatives, and the first class (in this case RVs) are considered positives and the second class (in this case NCSs) are considered negatives. The last value in the table (SVs) is the average percentage of vectors from the training data used as support vectors. These are the vectors which lie on the margin of the SVM classifier. More support vectors mean a more complex model.

The best classifiers used just the entropy features. These produced the simplest (fewest support vectors), most accurate models with nearly perfect sensitivity, specificity, and accuracy, regardless of the division of the data. The SVMs trained with all the features were also highly accurate, but they required a more complex model. The Fourier transform based features created less accurate classifiers whose accuracy depended much more on how the data was divided, particularly for the Fourier phase vector features. This suggests that there are anomalous data points in one or the other of these data sets, at least

Table 6.3: Results for HERV-NCS classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.99	1.00	0.98	1.00	0.98	1.00	12%
e1, e2, e3, e6	0.98	1.00	0.98	1.00	0.98	1.00	13%
mRY, mMK, mSW	0.71	0.86	0.72	0.85	0.72	0.82	57%
f(1), f(2), f(3:5), f(6:9), and avgblk	0.62	0.76	0.73	0.86	0.68	0.77	57%

in respect to these features.

6.2.4 Detecting HERVs

The problem of detecting HERVs was then addressed. First, the problem of distinguishing them from the NCSs was examined. The results are shown in Table 6.3. A multi-dimensional scaling of the data onto two dimensions from the 12-dimensional space used for this problem is shown in Figure 6.1. The projection strongly suggests that the data are not linearly separable. However, the SVMs do nearly as good a job with this problem as with the RV-NCS problem. As with the RV-NCS classification, the entropy features produce simple, accurate classifiers. For this problem, using all 12 features results in classifiers that are just as good as the entropy classifiers. The classifiers built using the Fourier transform based features only are again more complex and less accurate. However, note that the impact of choosing different divisions of the data is less than it was for the RV-NCS problem, suggesting the HERV data set is more uniform with respect to these features than the RV data set.

Then, the problem of distinguishing HERVs from genes (GENE data set) was ex-

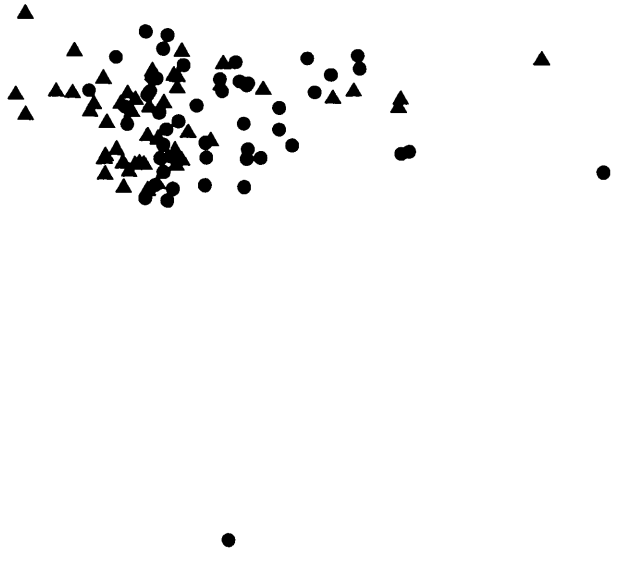


Figure 6.1: Multi-dimensional scaling of feature vectors representing HERVs and NCSs using all 12 features. HERVs are red triangles; NCSs are blue circles.

Table 6.4: Results for HERV-GENE classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.92	0.99	0.93	1.00	0.92	0.97	22%
e1, e2, e3, e6	0.88	0.96	0.92	0.99	0.90	0.96	24%
mRY, mMK, mSW	0.31	0.48	0.81	0.96	0.56	0.63	70%
f(1), f(2), f(3:5), f(6:9), and avgblk	0.92	0.99	0.91	0.99	0.91	0.96	30%

amined. This problem is more difficult than distinguishing HERVs from NCSs, because the genes contain both introns and exons, a mixture of coding and non-coding regions. The HERVs consist of coding regions with many mutations surrounded by non-coding regions. The results (shown in Table 6.4) are excellent for all feature subsets except the set of Fourier magnitude features. The Fourier phase vector classifiers are effective for this problem, achieving higher sensitivity than the classifiers built with the entropy features, although with a somewhat more complicated model. The classifiers using the three Fourier magnitude features have good specificity, but terrible sensitivity. This means these classifiers are good at identifying genes, but mistake retroviruses for genes more often than not.

A multiclass SVM with all 12 features that distinguished HERVs from GENEs from NCSs was trained. This SVM used 13% of the data vectors as support vectors and did perfect classification in the best case with average recall and precision values for all three classes of 0.94. (Precision measures the percentage of sequences assigned to a class that actually belong to it; recall measures the percentage of sequences belonging to a class that are, in fact, assigned to it.)

6.2.5 Distinguishing Different Types Of Viruses

The next group of experiments were done to see how well the features could distinguish different types of viruses. The first set of experiments distinguish HERVs from intact

Table 6.5: Results for HERV-RV classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.98	1.00	0.97	1.00	0.97	0.99	2%
e1, e2, e3, e6	0.92	1.00	0.92	1.00	0.92	1.00	49%
mRY, mMK, mSW	0.73	1.00	0.63	0.91	0.68	0.91	76%
f(1), f(2), f(3:5), f(6:9), and avgblk	0.81	1.00	0.74	1.00	0.78	1.00	53%

retroviral genomes (RVs). The difference between these two groups is that HERVs are heavily mutated. The results of this experiment are shown in Table 6.5. All feature sets except the Fourier magnitude subset produce good classifiers for this problem. The simplest and most accurate classifier is produced using all the features. This suggests that all the features are contributing significantly to solving the problem. This makes sense as all the features would be affected by mutation in different ways.

Figure 6.2 shows a visualization of the 12-feature vectors for the three data sets: HERV, RV, and NCS. The three data sets seem to fall into three natural groups. The black squares representing the non-coding NCSs are on the left of the figure; the blue circles representing the intact RVs are on the right, and the HERVs (red triangles) with mutated coding regions are in the middle.

The next set of experiments tested classifiers trained to distinguish lentiviruses (LENTI data set) from other types of retroviruses (NONLENTI data set). In this case only intact retroviral genomes were used. Lentiviruses are the genus of retroviruses that includes HIV. The main difference between lentiviruses and other retroviruses is that they have

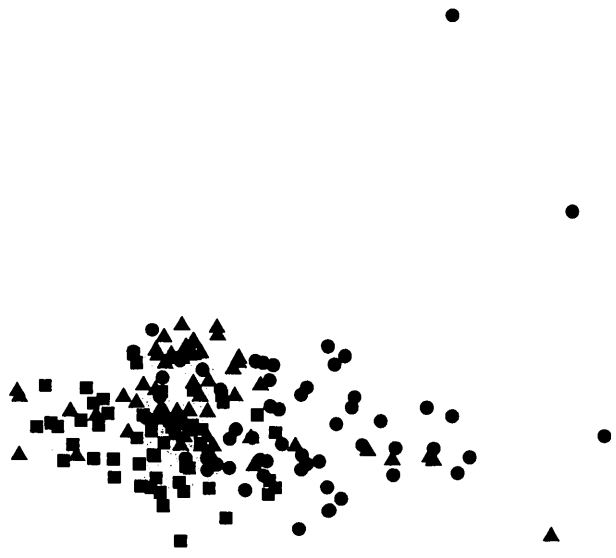


Figure 6.2: Multi-dimensional scaling of feature vectors representing HERVs, RVs, and NCSs using all 12 features. HERVs are shown as red triangles; NCSs are black squares; RVs are blue circles.

Table 6.6: Results for LENTI-NONLENTI classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.99	1.00	0.98	1.00	0.99	1.00	20%
e1, e2, e3, e6	0.98	1.00	0.99	1.00	0.99	1.00	24%
mRY, mMK, mSW	0.85	1.00	0.86	1.00	0.85	0.94	68%
f(1), f(2), f(3:5), f(6:9), and avgblk	0.99	1.00	1.00	1.00	0.99	1.00	18%

Table 6.7: Results for PAP-LENTI classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	1.00	1.00	1.00	1.00	1.00	1.00	8%
e1, e2, e3, e6	1.00	1.00	1.00	1.00	1.00	1.00	13%
mRY, mMK, mSW	1.00	1.00	0.96	1.00	0.98	1.00	10%
f(1), f(2), f(3:5), f(6:9), and avgblk	1.00	1.00	0.99	1.00	0.99	1.00	4%

some extra genes. The results of this classification are shown in Table 6.6. The best classifiers use the five phase vector features. Those classifiers get an average of 99% accuracy. The classifiers using all the features and the classifiers using just the entropy features are nearly as good. The classifiers based on the Fourier magnitude features have a trade-off between getting good sensitivity or good specificity. They also require many more support vectors.

The papilloma virus is not a retrovirus but is closely related. There are retroviruses

Table 6.8: Results for PAP-RV classification

Features	Sensitivity	Best	Specificity	Best	Accuracy	Best	SVs
All features	0.86	1.00	0.69	1.00	0.78	0.95	48%
e1, e2, e3, e6	0.84	1.00	0.88	1.00	0.86	1.00	52%
mRY, mMK, mSW	0.81	1.00	0.91	1.00	0.86	1.00	76%
f(1), f(2), f(3:5), f(6:9), avgblk	0.39	0.91	0.38	0.91	0.39	0.59	81%

that encode papilloma virus proteins. The papilloma virus structure [153] is different from that of a retrovirus. It is roughly the same length (8000 nucleotides) and consists of three regions: early (50%), late (40%), and a long control region (10%). The early region contains six ORFs; the late region contains two ORFs; the long control region does not encode proteins. Proteins are encoded using combinations of one or more ORFs. Like retroviruses, the ORFs in papilloma viruses lie in all three possible reading frames and sometimes overlap. The results for SVMs distinguishing papilloma viruses from lentiviruses are shown in Table 6.7. All groups of features produced accurate classifiers using a small number of support vectors. The simplest models were obtained using the Fourier phase vector features. The results were less good separating PAP from RV, the data set containing assorted retroviruses. These are shown in Table 6.8. For this problem, the Fourier phase vector features produced the worst classifier, and the entropy feature classifiers and Fourier magnitude classifiers produce the best, with the entropy classifiers using fewer support vectors. It is likely that the difficulty here is that the RV data set includes sequences that encode papilloma virus proteins or proteins similar to them, making the detectable difference between the two classes the non-coding long control region. The entropy features detect the randomness of this region, and the Fourier magnitude features detect that it is non-coding.

6.2.6 Conclusions About Use Of Statistical Features

In this section, twelve features were tested on various classification problems involving viral genomes, HERVs, and non-coding human genome sequences. All twelve features perform well, and they work well together. The results demonstrate that features designed for exon finding are useful for making much finer distinctions between sequences than just protein coding/not protein coding. The entropy features seem to contribute the most in classification problems involving NCS data set. This is likely because the most important distinction being made involves the randomness of the sequences. The Fourier phase vector features seem to contribute the most towards distinguishing different types of functional sequences. The Fourier phase magnitude features taken as a group do not excel over the other features in any of the experiments. However, since in many cases using all twelve features produced the best results, they contribute. It could be that a subset containing one or two features from each set would beat the performance of the feature subsets tested.

6.3 Classifiers Using SEMs Operating On ACGT Data

As an alternative approach, classifiers were built using SEM features. The advantage of SEM features over statistical features is that they do not need to be designed based on biological knowledge. Instead they have the potential to give biological insight. The

Algorithm 6: Build a DNA Sequence Classifier

Data: training data set *train*, test data set *test*, fitness function f, feature selection method m
Result: classifier, test result
$SEMset \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** 100 **do**
  Execute genetic algorithm using fitness function f;
  $SEMset \leftarrow SEMset \cup$ best SEM
end
$SEMtrain \leftarrow$ feature values of *train* for $SEMset$;
$SEMtest \leftarrow$ feature values of *test* for $SEMset$;
Select ten best features using method m and $SEMtrain$;
Build random forest classifier using selected features from $SEMtrain$;
Test on selected features from $SEMtest$;
return *classifier and test result*

classification problems studied using SEMs are described in Chapter 5: the sLTR/SINE problem (human sequences), the RT problem (fruit fly sequences), and the IES problem (*Tetrahymena* sequences). The disadvantage of using SEM features is that there are so many of them that feature selection becomes a central issue. Various different methods were explored for selecting good feature subsets. Since SEM features are a superset of k-mer features, classifiers were built based on k-mer features for comparison.

For these experiments, random forest classifiers were used. Classifiers were built using various feature subsets and tested on new data sets following Algorithm 6. Typically when string kernels are used, they are built into classifiers as a complete set, so classifiers built that way are included. The *randomForest importance* option in R scores the variables used according to how much the mean accuracy is decreased when that variable is omitted. Classifiers were built using the 10 most important *randomForest* features

(results in column headed "rF") and also using the 10 features with the highest information gain (results in column headed "IG"). Finally, classifiers were built using groups of approximately 10 features chosen with dissimilarity clustering (results in column headed DC – see Section 4.4.3). For the RT problem and the IES problem, the SEM classifiers all substantially outperformed the k-mer classifiers. For the sLTR/SINE problem, the two types of features produced classifiers of comparable quality, with the best classifier the complete string kernel.

It is possible that these results could be improved through parameter tuning. The parameters of the evolutionary algorithm (population size, number of mating events, mutation rate, number of states in SEMs, tournament size) could be changed, as could the feature selection parameters (number of replicates producing features, number of features in subsets, clustering method). The optimal values for these parameters are likely to be problem specific. In this thesis, the focus is on understanding the SEM features through comparisons between problems and, thus, common parameters are used without attempting to optimize them.

Each feature selection method has advantages independent of classification accuracy. The random forest importance method selects features that work best with the classifier; the information gain method selects features that are all individually good; DC selects a diverse set of features that classify well together.

Table 6.9: Accuracy of classifiers distinguishing solitary LTRs from SINEs on test data for random forests trained using SEM features with random forests trained using k-mer features using different types of feature selection.

features	all	rF	IG	DC
SEM features	97%	96%	96%	98%
k-mer features	100%	97%	96%	94%

6.3.1 Distinguishing SINEs From Solitary LTRs: sLTR/SINE Problem

The first classification problem is described in Section 4.3. It has two types of sequences: the long terminal repeat (LTR) portion of endogenous retroviruses (ERVs) and short interspersed nuclear elements (SINEs).

Table 6.9 shows the classification results. All of the classifiers in the table have high accuracies. The k-mer classifier that uses all the features gets slightly better results, and the k-mer classifier that uses features chosen by DC gets slightly worse results. This suggests that all k-mer features contribute to the classification and that, if it is desirable to reduce the size of the feature set, choosing the most effective k-mer features is a better strategy than choosing a diverse set. For SEM features, on the other hand, the best option seems to be a diverse set, though the difference is small enough that the result is not conclusive.

Table 6.10: Accuracy of classifiers distinguishing LTR retrotransposons, exons, and intergenic sequences on test data for random forests trained using SEM features with random forests trained using k-mer features using different types of feature selection.

features	all	rF	IG	DC
SEM	94%	93%	93%	94%
k-mer	88%	90%	89%	69%

6.3.2 Distinguishing LTR Retrotransposons, Exons, And Intergenic Sequences In *Drosophila*: RT Problem

Table 6.10 shows the classification results for the RT problem. For this problem, the SEM classifiers outperform the k-mer classifiers. All the SEM feature subsets produce classifiers with comparable performance. The k-mer classifiers for this problem using all the features, the randomForest features, and the information gain features are comparable to each other. The classifier built from features chosen using DC has a high error due to overfitting. For the k-mer features, it is counterproductive to choose a diverse set. Some of these features do not generalize well. Note that while SEM features undergo selection for quality in the genetic algorithm, k-mer features are generated with no quality selection. This means some are of poor quality.

6.3.3 Distinguishing IES From MDS Sequences In *Tetrahymena*: IES Problem

Table 6.11 shows the classification results for the IES problem. This classification problem is harder than either the sLTR/SINE classification problem or the RT problem. The

Table 6.11: Accuracy of classifiers distinguishing IESs from MDSs on test data for random forests trained using SEM features with random forests trained using k-mer features using different types of feature selection.

features	all	rF	IG	DC
SEM features	92%	92%	92%	91%
k-mer features	56%	88%	88%	68%

SEM classifiers all have comparable performance to each other. The k-mer classifiers using the features selected as best by either the randomForest importance function or the information gain function are effective and comparable to each other, but still less accurate than the SEM classifiers. The k-mer classifiers using all the features or those chosen by DC have an overfitting problem. Again, some k-mer features do not generalize well.

6.3.4 Feature Selection: Genomic vs. Consensus Sequences

In Section 5.2 the sLTR/SINE problem is studied using different sources for the training/testing data in order to demonstrate how the effective features can be analyzed to obtain biological insight about the sequences. Here, the effectiveness of the various feature selection techniques is examined. Three methods for choosing features are compared. The first method is that used in previous work on SEMs. Evolve SEMs with many states and use the features from those with best fitness. SEMs with 20 states are evolved and the best is chosen from 100 replicates. The other methods involve evolving SEMs with a small number of features (making them more interpretable), pooling the features from the best fitness SEMs, and then performing feature selection. Four-state and six-state

Table 6.12: Classification accuracy using all three types of data and feature sets generated by dissimilarity selection for training.

		RepeatMasker Sequences			Retrotector Sequences		
Training Set	Feature Set	LTR	SINE	overall	LTR	SINE	overall
Consensus seq.	20 centres	96%	100%	98%	**49%**	100%	75%
RepeatMasker seq.	20 centres	99%	100%	100%	**71%**	100%	85%
Retrotector seq.	20 centres	100%	99%	100%	**89%**	99%	94%
Consensus seq.	50 centres	95%	100%	97%	**49%**	100%	74%
RepeatMasker seq.	50 centres	100%	100%	100%	**45%**	100%	73%
Retrotector seq.	50 centres	100%	99%	100%	**99%**	99%	99%

SEMs and evolved and two sorts of feature selection, dissimilarity selection and dissimilarity clustering, are compared. Then, the classifiers are compared to classifiers created for the same purpose using other methods.

6.3.4.1 Dissimilarity Selection

Dissimilarity selection is useful both for feature selection and for getting a better understanding of the features and their properties. By clustering and selecting a representative from each cluster, a diverse set of features is obtained. The number of features that need to be analyzed in order to understand the feature set is reduced to the number of clusters. Figure 6.3 shows a visualization of the entire feature set with cluster centres marked with a © symbol. The figure shows some regions in which features close to each other are found by the genetic algorithm many times, but, on the whole, it demonstrates that a wide variety of features are being found.

Figure 6.5 is a visualization of how the data is separated using the 20 cluster centres.

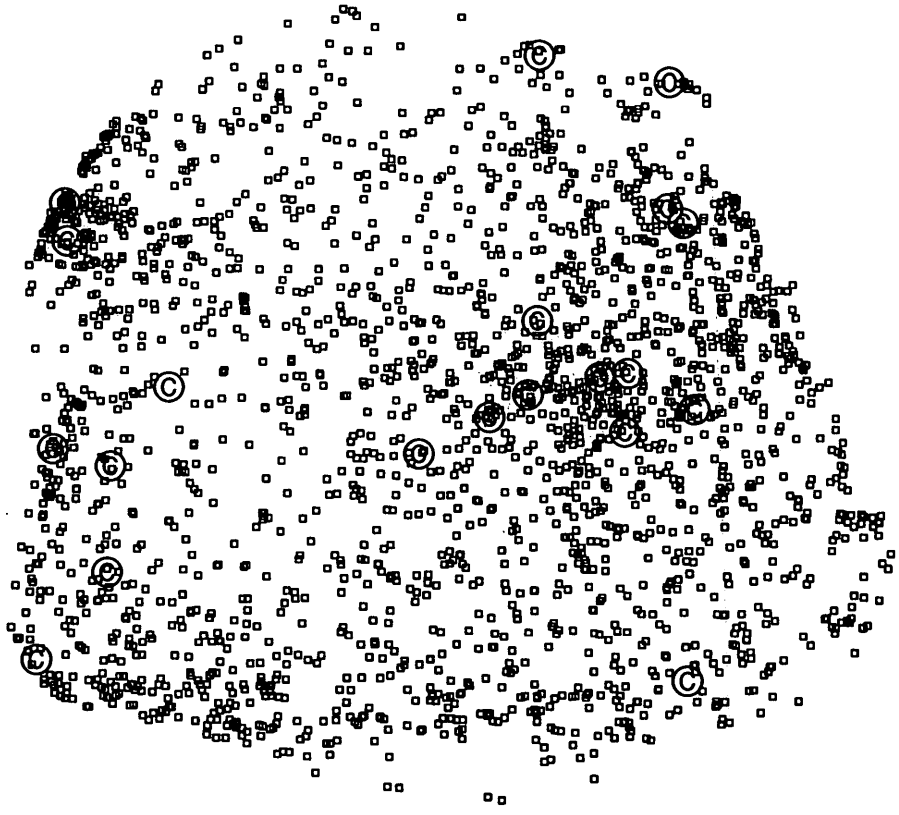


Figure 6.3: Depiction of feature absolute correlation distances using multi-dimensional scaling to display in two dimensions. Cluster centres are represented by red "©" symbols.

Table 6.13: Probability an evolved 20-state machine will create a classifier as good as these produced by DS.

Training Set	Feature Set	Repeatmasker Sequences probability	Retrotector Sequences probability
Consensus seq.	20 centres	0.23	0.02
RepeatMasker seq.	20 centres	0.57	0.16
Retrotector seq.	20 centres	0.00	0.91
Consensus seq.	50 centres	0.28	0.02
RepeatMasker seq.	50 centres	0.14	0.51
Retrotector seq.	50 centres	0.00	0.00

Table 6.14: Classification accuracy using "best" representative from each cluster.

Training set	Used to choose	RepeatMasker Sequences LTR	SINE	overall	Retrotector Sequences LTR	SINE	overall
Consensus seq.	RT data	97%	100%	99%	**52%**	100%	76%
RepeatMasker seq.	RT data	100%	99%	99%	**59%**	99%	79%
RetroTector seq.	RT data	100%	98%	98%	**100%**	99%	99%
Consensus seq.	RM data	99%	100%	99%	**51%**	100%	75%
RepeatMasker seq.	RM data	100%	100%	100%	**53%**	100%	76%
RetroTector seq.	RM data	100%	99%	99%	**100%**	100%	100%
Consensus seq.	RB data	99%	100%	99%	**61%**	100%	80%
RepeatMasker seq.	RB data	99%	100%	99%	**79%**	100%	89%
RetroTector seq.	RB data	100%	97%	99%	**99%**	99%	99%

Table 6.15: Probability an evolved 20-state machine will create a classifier as good as these produced by DS choosing the "best" representative from each cluster.

Training set	Used to choose best	RepeatMasker Sequences probability	Retrotector Sequences probability
Consensus seq.	RT data	0.05	0.01
RepeatMasker seq.	RT data	0.94	0.27
RetroTector seq.	RT data	0.67	0.00
Consensus seq.	RM data	0.05	0.02
RepeatMasker seq.	RM data	0.06	0.33
RetroTector seq.	RM data	0.10	0.00
Consensus seq.	RB data	0.05	0.00
RepeatMasker seq.	RB data	0.94	0.10
RetroTector seq.	RB data	0.10	0.00

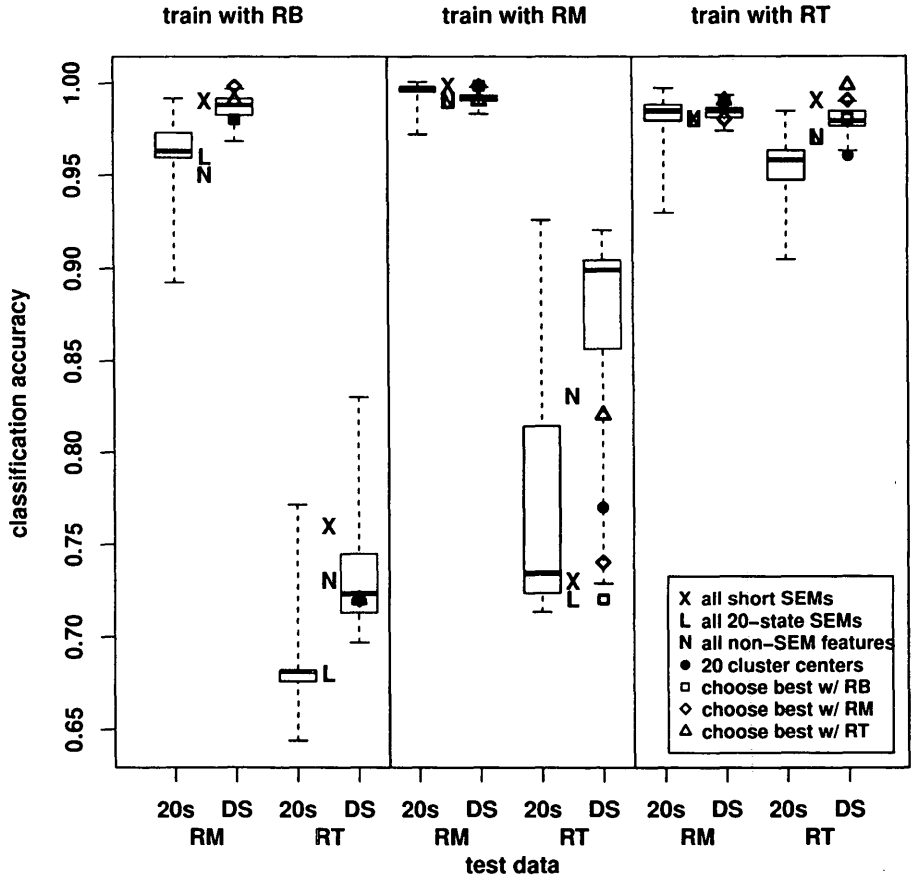


Figure 6.4: Accuracy of classifiers using different types of data sets for training and testing. Box plots represent the distribution of accuracies produced by classifiers created with individual 20-state evolved machines and with groups of 20 SEM features chosen by DS with random selection. Between the boxplots are shown the accuracies of classifiers built using all the 4- and 6-state SEM features (X), all the 20-state SEM features (L), and all the non-evolved features (N). Also shown as impulses are the accuracies of four classifiers created using DS with "center" and "best" selection methods.

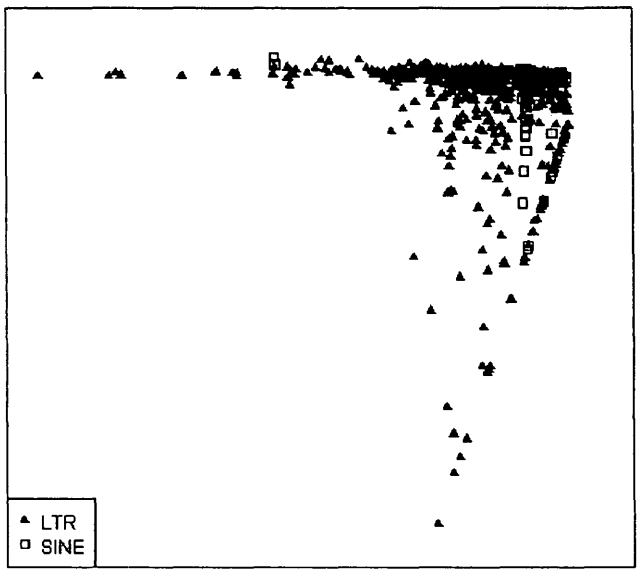


Figure 6.5: Projection into two dimensions of solitary LTRs and SINEs from all data sets represented using the 20 cluster centres. Notice that the SINEs, represented by the squares, group together.

It displays solitary LTRs and SINEs from the combined data sets represented by the 20 features generated by the cluster centres. This figure suggests that, although the classes are not linearly separable, the problem is doable, as the SINEs mostly cluster together in the upper right of the figure.

Figure 6.4 shows the results of classification using DS (see Section 4.4.2) along with boxplots showing the distribution of results for the 100 best evolved 20-state individual classifiers and 100 classifiers built with DS with random selection from 20 clusters from

the 4- and 6-state SEM features. The 20-state machines were evolved using the same genetic algorithm as for the 4- and 6-state machines. Also shown in the figure are the results of DS on the 4- and 6-state machines using four other selection methods, and the results of building classifiers using all the features with no selection for three groups: 20-state SEM features, 4- and 6-state SEM features, and the non-SEM features. All three types of data set are used for training and the two genomic data sets for testing.

Six combinations of train/test data are used: RB/RM, RB/RT, RM/RM, RM/RT, RT/RM, and RT/RT. The consensus sequences (RB) were just used for training since the intended applications were for genomic sequences. For four of these combinations every selection method obtained greater than 97% accuracy. For these combinations, all but one of the 20-state SEM classifiers achieved better than 90% accuracy, and all three groups of features with no feature selection created classifiers with better than 95% accuracy.

It is more challenging to create high accuracy classifiers for the other two problems, classifying RT sequences with classifiers trained on RB or RM sequences. Better classifiers can be created when training is done with genomic sequences than when done with consensus sequences. Most of the DS classifiers perform better than most of the 20-state SEM classifiers. The best classifier trained with RB data achieves 83% accuracy and was created with DS selecting from each cluster at random. The best classifier trained with RM data achieves 93% accuracy and was created using a 20-state SEM. For these

problems, the best selection method for DS is random selection, choosing the best of 100 feature sets. This demonstrates that choosing features that work well together is more important than choosing the *best* features or features that are maximally diverse. For both these problems, feature selection is worthwhile, yielding better classifiers than the one built from all the features.

The choice of 20 clusters was arbitrary, chosen to match the number of features in the 20-state SEMs. Increasing the number of clusters from 20 to 50 and performing DS with random selection does not change the median accuracy of classifiers trained on RM and tested on RT data, but decreasing the number of clusters to 10 reduces the median accuracy by 5%. Future work will examine this question in more detail to determine the optimal number of clusters.

6.3.5 Dissimilarity Clustering

137 classifiers were trained on diverse subsets of 2743 features that included the 4- and 6-state looping and non-looping SEMs as well as the non-SEM features described in Section 5.2.2. The feature sets ranged in size from 4 to 46 with a mean of 20. The results are shown in Figure 6.6 along with results for individual 20-state SEMs. They are trained with all three types of data and tested on a data set combining RM and RT data. For both types of classifier, training with RT data yields results that are both better and more consistent. In all but one case, classifiers created using DC get better than 82%

Table 6.16: Classification accuracy using all three types of data and 137 feature sets generated by dissimilarity clustering for training.

Training set	LTR range	LTR avg.	SINE range	SINE avg.
	RepeatMasker sequences			
Consensus Sequences	83% to 98%	92%	96% to 100%	100%
RepeatMasker Sequences	99% to 100%	100%	98% to 100%	100%
RetroTector Sequences	95% to 100%	99%	96% to 100%	99%
	Retrotector Sequences			
Consensus Sequences	38% to 84%	47%	80% to 94%	91%
RepeatMasker Sequences	43% to 90%	66%	98% to 100%	100%
RetroTector Sequences	85% to 99%	93%	95% to 99%	98%

accuracy. When trained on genomic sequences, they get better than 87% accuracy. The best classifiers get 95% accuracy trained on RB data, 98% accuracy trained on RM data, and 99% accuracy trained on RT data. For comparison with *rfcv* feature selection (see Section 4.4.1), The classifiers were retrained using data sets for training and testing that combined all three types of data. The best classifier had 99% accuracy with 10 features as compared to 92% accuracy with 343 features for *rfcv*.

Although most feature sets created classifiers with accuracies less than 90% when trained on the consensus sequence data, the best achieved an overall accuracy of 95% using four features. The same set of four features produced the best classifier when trained on RM data with an overall accuracy of 98%. Figure 6.7 shows a projection from four into two dimensions of the data using these four features.

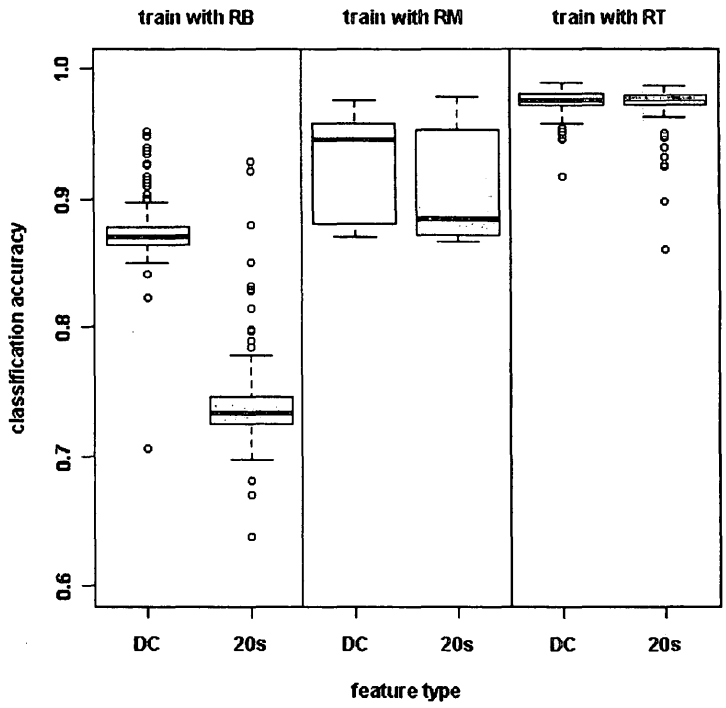


Figure 6.6: Classification accuracy of 137 feature sets generated by DC and 100 feature sets generated by individual evolved 20-state SEMs tested on mixed RM and RT data.

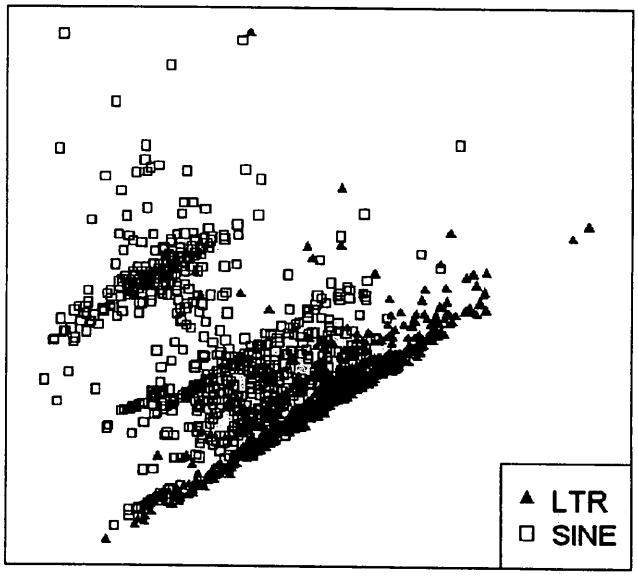


Figure 6.7: Projection into two dimensions of sLTRs and SINEs from all data sets represented using the four super-features.

Table 6.17: Comparison of results of the SEMclass classifier with TEclass, REPCLASS, and classifiers using k-mer features. Shown are percentages identified correctly (corr.), incorrectly (incorr.), or not identified (?).

Classifier	Retrotector sLTRs			RepeatMasker sLTRs		
	corr.	incorr.	?	corr.	incorr.	?
SEMclass-DS (best)	100%	0%	0%	100%	0%	0%
SEMclass-DC (best)	99%	1%	0%	100%	0%	0%
TEclass	76%	15%	9%	97%	2%	1%
k-mer	89%	11%	0%	100%	0%	0%
k-mer-DC	85%	15%	0%	94%	6%	0%
SEMclass-DS (consensus)	61%	39%	0%	99%	1%	0%
SEMclass-DC (consensus)	84%	16%	0%	98%	2%	0%
REPCLASS Structural	0%	3%	97%	0%	0%	100%
REPCLASS Homology	61%	9%	30%	96%	0%	4%

Classifier	RepeatMasker SINES		
	corr.	incorr.	?
SEMclass-DS (best)	100%	0%	0%
SEMclass-DC (best)	100%	0%	0%
TEclass	95%	3%	2%
k-mer	100%	0%	0%
k-mer-DC	99%	1%	0%
SEMclass-DS (consensus)	100%	0%	0%
SEMclass-DC (consensus)	100%	0%	0%
REPCLASS Structural	41%	0%	59%
REPCLASS Homology	1%	0%	99%

6.3.6 Using DS On Non-SEM Features

Fig 6.4 shows the results of classifiers built with the entire set of non-SEM features on each of six problems with an "N." For the four easier problems, these classifiers achieve accuracies $> 95\%$. For the two harder problems, feature selection is of benefit. When trained on RB data, a classifier using all the non-SEM features achieves 73% accuracy on the RT data; the best classifier using DS on non-SEM features with random selection achieves 80% accuracy (best SEM classifier got 83%). When trained on RM data, the all-feature non-SEM classifier gets 83% accuracy, and the best DS classifier achieves the same accuracy as the best DS SEM classifier, 93%.

6.3.7 Comparison With Other Methods

Table 6.17 compares the performance of two classifiers created by other researchers, TEclass and REPCLASS (see Section 2.2.6), with the best classifiers based on DC and DS, referred to as SEMclass-DC (best) and SEMclass-DS (best) respectively. The algorithms are tested on the solitary LTRs generated by Retrotector and Repeatmasker and the SINEs generated by Repeatmasker.

TEclass performs comparably to both SEMclass classifiers on the RM data, but does less well on the RT data. This is probably due to the fact that TEclass uses consensus sequences for training. To support this hypothesis the best results using consensus sequences for training are included, SEMclass-DS (consensus) and SEMclass-DC (con-

sensus). Also included are the accuracies of random forest classifiers using the 4-mer and 5-mer features used in TEclass but trained with RT sequences, one without any feature selection (k-mer) and one with features selected using DC (k-mer-DC).

On the RT solitary LTRs, TEclass has performance intermediate between SEMclass-DS (consensus) and SEMclass-DC (consensus). The k-mer classifiers using the TEclass features and trained on genomic features perform much better than TEclass, suggesting it could be improved by using genomic sequences for training. Note that feature selection degrades the performance of the k-mer classifier. This is consistent with the results in [2]. This means that k-mer features could not be used to fulfil the goal of creating a comprehensible classifier.

As expected, since it was not designed for the purpose, the REPCLASS Structural module performs poorly on the solitary LTR data. It identifies about half the SINE elements correctly by detecting short sequence repeats or poly-A tails.

The REPCLASS Homology module was expected to do better than it did on the data generated by RepeatMasker, as it was using the same database to identify the sequences as RepeatMasker did to generate them. The reason it performed poorly is that RepBase was designed to be human readable, not machine readable. REPCLASS found many matches that did not lead to a classification because RepBase did not include class information in the data record. A biologist may know immediately that *AluJo*, for example, is a SINE element, but REPCLASS does not. This problem occurs more often for SINE

elements than for solitary LTRs. Note that nearly a third of the RT solitary LTRs are labelled "unsure" by the Homology module, meaning they are probably not in RepBase. This demonstrates that annotations of solitary LTRs in the human genome are far from complete.

The REPCLASS TSD module is designed to run on "relatively small genomes." Because the computational power required to run it on the human genome was not readily available, it is not included in the comparison. SINE elements should have TSDs; solitary LTRs should not, so it would be expected to do a good job of distinguishing the two classes.

6.4 Classifiers Using SEMs Operating On Reading Frame Data

Based on the good results obtained by the statistical feature classifiers using reading frame data, it was decided to train SEMs whose transitions were driven by reading frame data rather than by the {A,C,G,T} alphabet. The same Fourier phase histogram was used as for the Fourier phase vector features (Section 3.2.1). The reading frame with the most members is designated Reading Frame 0; Reading Frame 1 is shifted one nucleotide from Reading Frame 0; Reading Frame 2 is shifted two nucleotides. A single string is constructed from the reading frame information gleaned from sliding a window across each of the three binary strings, by combining them base 3. For example, 5 is 012 base 3. A 5 in the string means that String 0 signals it is in Reading Frame 2, String 1

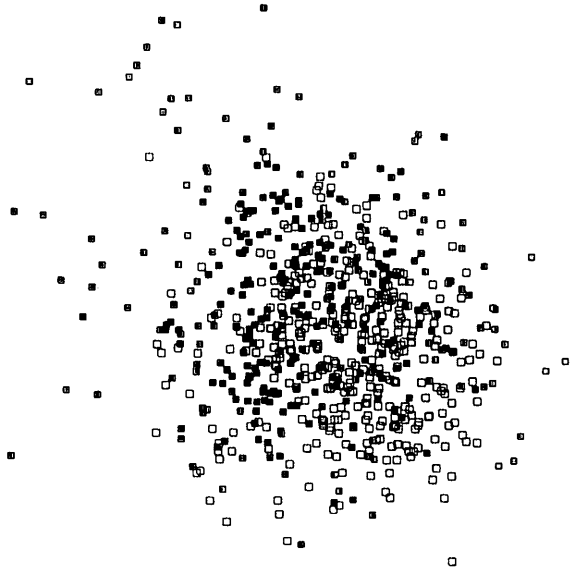


Figure 6.8: Projection from 10 dimensions onto 2 dimensions of clustering of HERV and NCS data sets using the original design, evolution with a changing neighbour set. HERVs are shown in black; NCSs in grey.

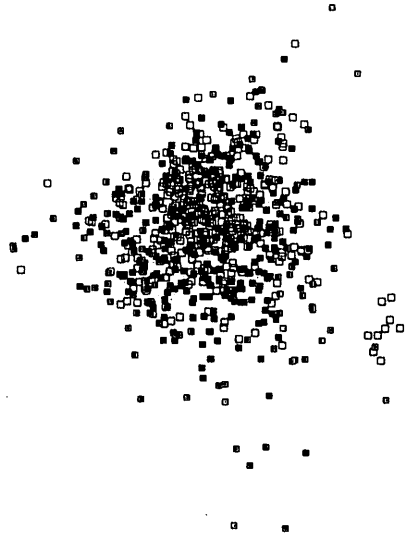


Figure 6.9: Projection from 10 dimensions onto 2 dimensions of clustering of HERV and NCS data sets using SEMs evolved with a coevolving neighbour set. HERVs are shown in black; NCSs in grey.

Table 6.18: Results of experiments on HERV and NCS data sets using the original design and a changed design with a coevolving neighbour set. Results using k nearest neighbour classification (knn) and SVM classification for the best replicate and averages are shown.

	Sens.	Spec.	Acc.
changing neighbours during training			
Knn Best	0.75	0.63	0.69
Knn Best	0.52	0.85	0.69
SVM Best	0.85	0.79	0.82
Knn Avg.	0.58	0.73	0.66
SVM Avg.	0.73	0.69	0.71
coevolving neighbours			
Knn Best	0.76	0.84	0.80
SVM Best	0.82	0.90	0.86
Knn Avg.	0.66	0.77	0.72
SVM Avg.	0.74	0.68	0.71

signals it is in Reading Frame 1, and String 2 signals it is in Reading Frame 0. Mixed signals are common in non-coding or overlapping regions. SEMs can be driven using this information either with three transitions (using a single indicator sequence, such as RY), with nine transitions (using two indicator sequences), or 27 transitions (using all three indicator sequences).

Ten-state SEMs were evolved using a knn fitness function. No feature selection was done. One hundred classifiers were created using the best SEM from each of 100 evolutionary replicates. Knn classifiers were compared with SVM classifiers.

The first experiment was done using the HERV and NCS data sets with $k = 60$ using the RY and SW strings to drive 9 transitions in the SEMs. For each replicate, calculations were done for sensitivity, specificity, and prediction accuracy. HERVs are

Table 6.19: Results of experiments using SEMs trained on individual strings and all together. Results using k nearest neighbour (knn) and SVM classification for the best replicate and averages are shown.

	Sens.	Spec.	Acc.
RY string only			
Knn Best	0.75	0.66	0.71
SVM Best	0.86	0.62	0.74
Knn Avg.	0.63	0.63	0.63
SVM Avg.	0.74	0.51	0.62
MK string only			
Knn Best	0.78	0.59	0.68
SVM Best	0.70	0.75	0.73
Knn Avg.	0.60	0.61	0.60
SVM Avg.	0.67	0.52	0.60
SW string only			
Knn Best	0.76	0.79	0.78
SVM Best	0.76	0.86	0.81
Knn Avg.	0.64	0.72	0.68
SVM Avg.	0.74	0.61	0.68

	Sens.	Spec.	Acc.
RY and SW strings			
Knn Best	0.76	0.84	0.80
SVM Best	0.82	0.90	0.86
Knn Avg.	0.66	0.77	0.72
SVM Avg.	0.74	0.68	0.71
all strings			
Knn Best	0.75	0.85	0.80
SVM Best	0.85	0.75	0.80
Knn Avg.	0.67	0.80	0.80
SVM Avg.	0.74	0.63	0.69

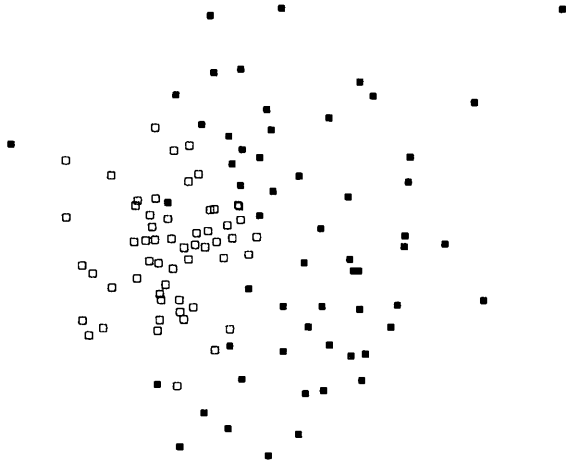


Figure 6.10: Projection from 10 dimensions onto 2 dimensions of clustering of RV and NCS data sets. RVs are shown in black; NCSs in grey.

considered positives and NCSs are considered negatives. The best of these and averages were determined for the best SEMs from each replicate using the complete data sets for k nearest neighbours and the test data from 5-fold cross validation for the SVMs. A visualization of the vectors produced can be seen in Figure 6.8. It shows a multi-dimensional scaling of the 10-dimensional data generated by the SEM with the highest RAND index onto two dimensions. Examination of this figure shows that the SEM did not do a good job of producing vectors which separate the two categories. Using k nearest neighbours classification, the average prediction accuracy was only 66% and the best of the hundred replicates only achieved 69% accuracy. The SVM classifier was able to boost this to an average of 71% accuracy with a best of 82%.

Table 6.20: Training Results for HERV and RV data sets. Results using k nearest neighbour (knn) and SVM classification for the best replicate and averages are shown.

	Sens.	Spec.	Acc.
HERV data set			
Knn Best	0.76	0.84	0.80
SVM Best	0.82	0.90	0.86
Knn Avg.	0.66	0.77	0.72
SVM Avg.	0.74	0.68	0.71
RV data set			
Knn Best	1.00	1.00	1.00
SVM Best	1.00	1.00	1.00
Knn Avg.	0.95	0.98	0.97
SVM Avg.	0.94	0.76	0.85

Because of the poor performance of the k nearest neighbour classifier, the experimental design was modified to try to get better separation. Instead of changing the neighbour set randomly during evolution, a neighbour set was allowed to co-evolve with the SEM. Every hundred mating events, a point in the neighbour set was swapped with a point not currently being used for training. If the average fitness was unchanged or improved, this new neighbour set was used. Otherwise, the algorithm continued to use the old neighbour set. Also, k was reduced to 10. The neighbour set was then saved along with the SEM produced. The k nearest neighbour results of this experiment were better with an average prediction accuracy of 72% with the best replicate predicting with an accuracy of 80%. The SVM, however, got similar results with both experimental designs. The results of these two experiments are shown in Table 6.23. The table shows sensitivity, specificity, and prediction accuracy for the best replicate and the averages. Some replicates have a

Table 6.21: Results of experiments classifying HERV data with RV SEMs and RV data with HERV SEMs. Results using k nearest neighbour (knn) and SVM classification for the best replicate and averages are shown.

	Sens.	Spec.	Acc.
HERV with RV classifier			
Knn Best	0.54	0.79	0.67
SVM Best	0.86	0.79	0.82
Knn Avg.	0.46	0.79	0.62
SVM Avg.	0.72	0.66	0.69
RV with HERV classifier			
Knn Best	0.91	0.84	0.88
SVM Best	1.00	1.00	1.00
Knn Avg.	0.76	0.76	0.76
SVM Avg.	0.89	0.75	0.83

sensitivity value that is much higher than the specificity value or vice versa. For example, there was a tie for best k nearest neighbour classifier in the first experiment. Of the two replicates which achieved 69% accuracy, one replicate was better at identifying HERVs with a sensitivity of 75% and a specificity of 63%; the other was better at identifying NCSs with a specificity of 85% and a sensitivity of only 52%. The projection in Figure 6.8 shows the vectors created by the SEM which is better at identifying HERVs (higher sensitivity). A multi-dimensional scaling of the vectors created by the best SEM in the second experiment is shown in Figure 6.9. The clusters are still not clearly defined, but it does appear that there is better separation in Figure 6.9 than in Figure 6.8. Since this is a projection from 10 dimensions to two, there could be better separation than there appears to be.

The next set of experiments were done to better understand the contributions of each of three strings (RY, MK, and SW). The RY and SW strings were used in the first set of experiments, because they usually carry the most biological meaning. The MK string was not used because using 27 transitions instead of 9 increases the risk of overtraining and also because it increases the risk that there will be transitions in the machine that were untested during the course of evolution and hence meaningless. The predictive value of each string was tested separately, and also the predictive value of all strings used together. The results are shown in Table 6.24. The string with the best predictive value alone using k nearest neighbour classification was the SW string. The best SEM was able to predict 78% of the values correctly, and on average the SW SEMs had a predictive accuracy of 68%. The RY string was next best with a best predictive accuracy of 71% and an average of 63%. The MK string came in last, as expected, with a best predictive accuracy of 68% and an average of 60%. The SVMs were not able to significantly improve the results using single strings. There was less variation, and the best values were somewhat higher. When the strings were used all together, the k nearest neighbour classifier achieved a best predictive accuracy of 80% which equals the best achieved by the RY and SW string pair. In this case, the SVM performance was lower with an average prediction accuracy of 69% and a best of 82%. However, that the SVM results are reported only on test data while the k nearest neighbour results include the training data. The poor performance of the SVM in this case confirms the hypothesis that 27 transitions is too many. The results

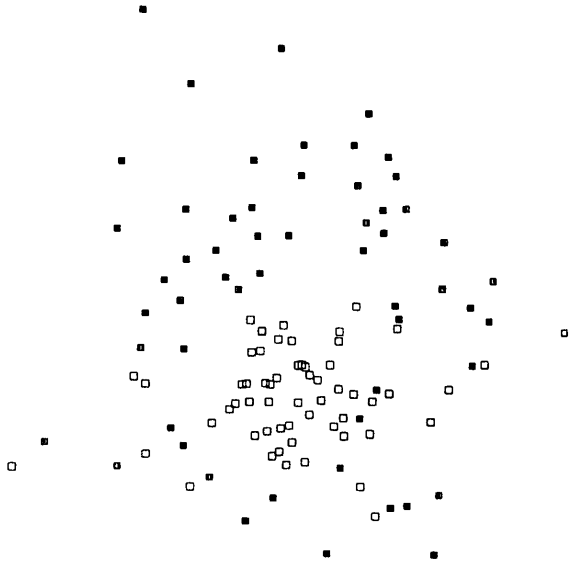


Figure 6.11: Projection from 10 dimensions onto 2 dimensions of clustering of RV and NCS data sets using SEMs trained to distinguish HERVs from NCSs. RVs are shown in black; NCSs in grey.

do not generalize well to unseen data.

Note that, using k nearest neighbours, the average sensitivity and specificity values for the RY and MK string machines are similar, but the SW SEMs have significantly higher specificity. This suggests the SW phase information is more indicative of not being a HERV than of being a HERV. These values vary, of course, for individual machines. The best machines for both the RY SEMs and the MK SEMs have higher sensitivity than specificity, and the best machine for the SW SEMs has close to equal sensitivity and specificity.

The next set of experiments was done using the RV data set of complete retroviral genomes using the RY and SW strings (9 transitions). These were much easier for the

Table 6.22: Results of experiments distinguishing lenti retroviruses from non-lenti retroviruses. Results using k nearest neighbour (knn) and SVM classification for the best replicate and averages are shown.

	Sens.	Spec.	Acc.
HERV with RV classifier			
Knn Best	1.00	0.98	0.99
SVM Best	1.00	1.00	1.00
Knn Avg.	0.96	0.97	0.97
SVM Avg.	0.90	0.96	0.93

SEMs to classify than the HERVs were. A multi-dimensional scaling of the resulting vectors is shown in Figure 6.10 and the results in Table 6.25. It is clear that the data is well separated. The best SEM was able achieve perfect classification using k nearest neighbours, and the average machine predicted 97% correctly. The SVMs also achieved perfect classification with an average of 85%.

The next set of experiments tested the ability of the classifiers to distinguish different types of retroviruses, lentiviruses and non-lentiviruses. The classifier was easily able to distinguish these two different sorts of retroviruses. A multi-dimensional scaling of the clustering is shown in Figure 6.12 and the classification results in Table 6.22.

The final set of experiments tested the SEMs on data sets for which they were not trained. The original thought was that training the SEMs on complete genomes would cut out the "noise" caused by mutations and classify based on the essential character of the retroviruses. In fact, the results were similar to those for the SEMs trained on the HERV data set. k nearest neighbour classifiers using the RV SEMs that were used to predict

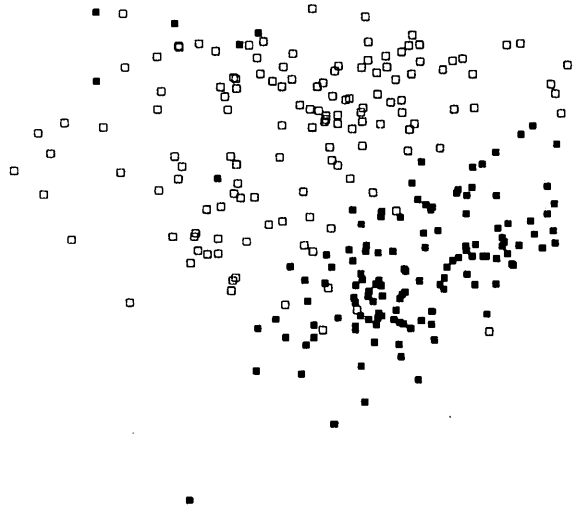


Figure 6.12: Multi-dimensional scaling from 10 dimensions onto 2 dimensions of clustering of LENTI and NLENTI data sets. LENTIs are shown in black; NLENTIs in grey.

RVs with nearly perfect accuracy predicted HERVs with an average of 62% accuracy (best 67%). The SVMs boosted this to an average of 69% with a best of 80%, very similar to their performance using the HERV classifiers. Classifiers using SEMs trained on the HERV data set also had similar performance predicting RVs to the classifiers using SEMs trained on the RV data set. Using k nearest neighbours, they predicted them with an average accuracy of 76% (best 88%), better than they did predicting the HERV they were trained on. The SVMs were able to achieve perfect classification in the best case with an average of 83% prediction accuracy. The results of these experiments are shown in Table 6.26, and a multi-dimensional scaling of the RV data vectors using the best HERV SEM is shown in Figure 6.11.

6.4.1 Interpreting The SEMs

In the course of this work, a discovery was made that, even though the SEMs normalize for the length of the string, it is important that both data sets have the same distributions of lengths. If they do not, the SEMs use length as a distinguishing feature, something which was not wanted in this case. A transient state in the SEM that is accessed once and only once for every string can be used to measure length. Its value will always be $\frac{1}{n}$ where n is the string length.

The machine in Figure 4.3 is a 4 state machine evolved with the HERV data set. With nine transitions from each state, it is difficult to interpret. However, it is not utterly obscure. Note that self-loops function as counters of blocks of certain values. State 0 has a self-loop for the values 0 and 4. Zero indicates both strings agree on reading frame 0; four indicates both agree on reading frame 1. State 0 is counting blocks of reading frames 0 and 1. State 1 has a self-loop if one or both strings indicate reading frame 2. So, it is counting blocks in reading frame 2. States 2 and 3 self-loop when the two strings indicate different reading frames. They could be keeping track of sequences which indicate overlapping regions.

Table 6.23: Results of experiments on HERV and NCS data sets using original design and changed design which fixes the neighbours.

	RAND	Sens.	Spec.	Acc.
changing neighbours during training				
Best	0.57	0.75	0.63	0.69
Best	0.57	0.52	0.85	0.69
Average	0.55	0.58	0.73	0.66
fixing neighbours				
Best	0.68	0.76	0.84	0.80
Average	0.62	0.66	0.77	0.72

Table 6.24: Results of experiments using SEMs trained on individual strings and all together.

	RAND	Sens.	Spec.	Acc.
RY string only				
Best	0.58	0.75	0.66	0.71
Average	0.56	0.63	0.63	0.63
MK string only				
Best	0.57	0.78	0.59	0.68
Average	0.54	0.60	0.61	0.60
SW string only				
Best	0.66	0.76	0.79	0.78
Average	0.59	0.64	0.72	0.68
RY and SW strings				
Best	0.68	0.76	0.84	0.80
Average	0.62	0.66	0.77	0.72
all strings				
Best	0.68	0.75	0.85	0.80
Average	0.63	0.67	0.80	0.80

Table 6.25: Training Results for HERV and RV data sets.

	RAND	Sens.	Spec.	Acc.
HERV data set				
Best	0.68	0.76	0.84	0.80
Average	0.62	0.66	0.77	0.72
RV data set				
Best	1.00	1.00	1.00	1.00
Average	0.94	0.95	0.98	0.97

Table 6.26: Results of experiments classifying HERV data with RV SEMs and RV data with HERV SEMs.

	RAND	Sens.	Spec.	Acc.
HERV with RV classifier				
Best	0.56	0.54	0.79	0.67
Average	0.53	0.46	0.79	0.62
RV with HERV classifier				
Best	0.78	0.91	0.84	0.88
Average	0.67	0.76	0.76	0.76

Table 6.27: Classification results using SVMs. Averages are shown.

Experiment	Sens.	Spec.	Acc.
HERV/NCS using RY and SW with original design	0.73	0.72	0.71
Best	0.85	0.83	0.81
HERV/NCS using RY and SW with fixed neighbours	0.74	0.73	0.71
Best	0.86	0.83	0.79
HERV/NCS using RY string only	0.74	0.67	0.62
Best	0.94	0.89	0.73
HERV/NCS using MK string only	0.67	0.63	0.59
Best	0.96	0.88	0.67
HERV/NCS using SW string only	0.74	0.71	0.68
Best	0.92	0.87	0.80
HERV/NCS using all strings	0.74	0.71	0.68
Best	0.89	0.81	0.75
RV/NCS using RY and SW with fixed neighbours	0.93	0.92	0.86
Best	1.00	1.00	1.00
RV/NCS using HERV/NCS classifier	0.89	0.88	0.83
Best	1.00	1.00	1.00
HERV/NCS using RV/NCS classifier	0.72	0.70	0.68
Best	0.86	0.82	0.80

6.5 Conclusion

In this chapter the use of the features described in Chapters 3 and 4 was demonstrated in classifiers. SVM, random forest and k nearest neighbour classifiers were used to distinguish HERVs, exogenous retroviruses, intergenic sequences, and genes, as well as to distinguish various types of viruses from each other. The sLTR/SINE problem from Chapters 4 and 5 was revisited, showing that SEM features could be used to distinguish solitary LTRs from SINEs with high accuracy. The feature selection techniques were demonstrated and compare to the method used in previous work of evolving a single SEM with many states to generate features for a classifier. This problem was studied using a variety of different data sets (both genomic and consensus) for training and testing. Finally, the use of SEMs other than with $\{$A,C,G,T$\}$ transitions was explored, and it was demonstrated that they can be used to recognize patterns of reading frame use in retroviruses.

The next chapter will incorporate some of these classifiers into a scanner that scans a genome and identifies the ERV sequences on it.

7 Scanning Genomes

The topic of this chapter is the on-going development of a software tool called LTRsieve that scans genomes to identify LTR retrotransposons. LTRsieve is tested on the five major chromosomes of the *Drosophila melanogaster* (fruit fly) genome and on *Homo sapiens* chromosome 21. LTRsieve is not meant to replace other methods, like RepeatMasker and RetroTector, but rather to supplement them. For genome annotation, different approaches produce different results, and the best annotations result from using a variety of tools. LTRsieve uses the statistical features discussed in Chapter 3 together with a random forest classifier.

The *Drosophila melanogaster* genome was used to test the accuracy and completeness of the tool, since *Drosophila melanogaster* has annotations for LTR retrotransposons based on several bioinformatics tools as well as manual annotation and thus provides a baseline in which one can have some confidence. It was possible to identify LTR retrotransposons on the *Drosophila melanogaster* genome with high accuracy. In addition, it was possible to check whether LTRsieve was sensitive to the choice of training data, in particular to whether the training data was based on data from *Drosophila* or on data

from other species. This is important for determining its usefulness as a generalized tool. To get an idea of how LTRsieve's functionality extended to other organisms, Chromosome 21 from the *Homo sapiens* reference genome was used for testing. The problem is more challenging for *Homo sapiens* than for *Drosophila melanogaster*, not only because the *Homo sapiens* genome is much larger, but also because the ERVs in *Homo sapiens* were inserted longer ago and thus have more mutations, making them harder to recognize. It is also more difficult to evaluate results as the annotations are not as thorough or consistent as those for *Drosophila melanogaster*. Results are compared to the results of RepeatMasker, which uses sequence homology to ERVs in RepBase to find fragments of ERVs, and to the results of RetroTector, a program which finds complete ERVs based on various sequence motifs. There is agreement from both RepeatMasker and RetroTector on some of the sequences identified as LTR retrotransposons by LTRsieve. There are also some additional putative newly identified ERVs.

7.1 Approach

LTRsieve operates by extracting sequences from a long genomic sequence using a sliding window together with a random forest classifier to determine whether the sequence is likely to be part of an LTR retrotransposon. When it finds many putative LTR retrotransposon windows in a row, it tests for LTRs at the beginning and end of the group using sequence alignment.

Table 7.1: Algorithm parameters with values used here.

name	use	value
w_{min}	minimum number of windows to test	3
f	amount of flanking DNA to examine	3000
i_{min}	minimum identity for LTRs	70%
d_{min}	minimum length of LTR	50
N	window size	2400
s	slide length	120

7.2 Methods

LTRsieve operates on long genomic sequences (i.e. assembled chromosomes or contigs). It scans the sequence in both the sense and antisense directions using a sliding window of length N which slides by skipping ahead s base pairs, extracts features, and tests them using a random forest classifier for pieces of LTR retrotransposons. Whenever a group of more than w_{min} sequential windows that are potentially pieces of LTR retrotransposons is found, f bp of flanking DNA is added to each end, and the first half is aligned with the second half using the Smith-Waterman algorithm in order to identify potential LTRs. If there is a match with greater than i_{min} identity of at least length d_{min}, then the LTRs and the sequence between them is identified as a hit.

The random forest algorithm for this study uses five types of features. These features are described in detail in Chapters 2 and 3. The features include the two feature sets inspired by work done in gene finding, entropy features (4 features listed in Table 7.2) and DFT magnitude features at frequency $\frac{1}{3}$ (3 features listed in Table 7.3), together with

Table 7.2: Feature Set I used in LTRsieve

name	description
e1	entropy of single bases
e2	entropy of dimers
e3	entropy of trimers
e6	entropy of hexamers

Table 7.3: Feature Set II used in LTRsieve

name	description
mRY	mag. of Fourier coeff. $S(\frac{N}{3})$ in RY string
mMK	mag. of Fourier coeff. $S(\frac{N}{3})$ in MK string
mSW	mag. of Fourier coeff. $S(\frac{N}{3})$ in SW string

the three novel feature sets: the feature set that detects the use of overlapping reading frames (4 features listed in Table 7.4), the feature set that uses the distribution of DFT phase values to measure how the sequence differs from a random sequence (3 features listed in Table 7.5), and the feature set that uses DFT phase values to detect the extent to which the sequence is using multiple reading frames (21 features listed in Table 7.6). All of these features are fast to calculate. Running under Ubuntu using a 2.00 GHz processor, LTRsieve takes about a minute to process a million base pairs. Other features could be added to or substituted for these in future work. The development of SEM features in Chapter 4 and the exploration of their properties in Chapter 5 was motivated by a desire to incorporate them into LTRsieve.

The random forest classifier was chosen because it is a highly accurate classifier that can deal well with feature sets, such as mine, that have features with variable degrees of importance, because it does not require balanced data sets for training, because it can

Table 7.4: Feature Set III used in LTRsieve

name	description
$f(1)$	freq. of distance 1 between change points
$f(2:5)$	sum of freq. of distances 2, 3, 4, and 5 between change points
$f(>5)$	freq. of distances > 5 between change points
avgblk	avg. distance betw. change points

Table 7.5: Feature Set IV used in LTRsieve

name	description
χ^2RY	similarity to random histogram for RY
χ^2MK	similarity to random histogram for MK
χ^2SW	similarity to random histogram for SW

Table 7.6: Feature Set V used in LTRsieve

name	description
varRY-1,2,3	variance of RY histogram values for reading frame 1,2,3
varMK-1,2,3	variance of MK histogram values for reading frame 1,2,3
varSW-1,2,3	variance of SW histogram values for reading frame 1,2,3
varvarRY	variance of variance of RY histogram values
varvarMK	variance of variance of MK histogram values
varvarSW	variance of variance of SW histogram values
nRY-1,2,3	proportion of sequence in reading frame 1,2,3 for RY
nMK-1,2,3	proportion of sequence in reading frame 1,2,3 for MK
nSW-1,2,3	proportion of sequence in reading frame 1,2,3 for SW

easily do a three-way classification, and because of the potential that future analysis of the decision trees will result in greater understanding of the problem.

Training sequences for the random forest classifier are the same length as the window size N. They are assigned to three classes: LTR retrotransposons, exons, and intergenic sequences. LTRsieve was tested with several different training sets. The random forest tags a sequence as a potential LTR retrotransposon if at least 40% of the decision trees so classify it. (Since there are three categories, anything greater than 33% is suggestive.) The random forest had an OOB error of 0.05 when training.

In order to evaluate the results, only annotated LTR retrotransposons with two LTRs were used. This means that some LTR retrotransposon annotations were ignored. The annotations include sequences that have sequence homology with reference sequences of LTR retrotransposons but are missing one or both LTRs. LTRsieve would not be able to identify those, as it only identifies sequences with both LTRs. Sensitivity and specificity were calculated based on the number of overlapping bases in the identified LTR retrotransposons and the comparison set.

7.3 *Drosophila melanogaster*

LTRsieve was tested on all the major chromosomes in *Drosophila melanogaster*. Results were compared using five different sets of training data. The results for the major Drosophila melanogaster chromosomes are shown in Table 7.7. Sensitivity and speci-

ficity are calculated based on the number of overlapping base pairs in the annotated set of LTR retrotransposons and in the set of LTR retrotransposons identified by LTR-sieve. Since LTRsieve only identifies complete LTR retrotransposons (those with two LTRs), sensitivity and specificity were calculated comparing only to annotations with two LTRs. The specificity calculation assumes that all LTR retrotransposons in *Drosophila melanogaster* have been identified, which means it is likely an underestimate.

The first training set was generated by scanning the X chromosome in the sense direction and generating features just as they are generated for testing. A sliding window was classified as a

1. LTR retrotransposon,
2. exon, or
3. neither

if at least 80% of the bases in the window fell into that category in the annotations downloaded on FlyBase, release 5.29 [134]. Three classes of sequence were used: LTR retrotransposons, exons, and other, meaning that 80% of the sequence was not annotated as an exon or as a TE (of any type). Only sense strand annotations were used for the LTR retrotransposon and exon classes, but annotations in both directions were used for the other class. Windows falling in each category were chosen at random to create 1894 LTR retrotransopons samples, 2241 exon samples, and 1899 other samples. The random

Table 7.7: Results for *Drosophila melanogaster* using training data generated by scanning the X chromosome.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	126	0.903	0.997
2L	23	90	101	0.889	0.997
2R	21	92	111	0.823	0.995
3L	25	92	114	0.824	0.995
3R	28	85	93	0.962	0.999

Table 7.8: Results for *Drosophila melanogaster* using a small set (1417) of training data generated by scanning the X chromosome.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	120	0.861	0.996
2L	23	90	104	0.880	0.997
2R	21	92	129	0.756	0.993
3L	25	92	141	0.780	0.994
3R	28	85	97	0.949	0.999

forest classifier trained with these samples had an 5% OOB error.

To test the sensitivity of the method to the training examples used, a smaller set of training examples was used – one quarter the size. This training set had 1417 training examples, selected at random from the original set: 475 LTR retrotransposons, 474 exons, and 475 other. This yielded a higher training error (7% instead of 5%) and worse performance with an average sensitivity of 84.5%, down from 88.0%. Full results are shown in Table 7.8.

Since this method of gathering training data requires a genome as well annotated as *Drosophila*, which is not usually available, another test was done using a training set generated from collected sequences of LTR retrotransposons and exons from FlyBase.

Table 7.9: Results for *Drosophila melanogaster* using a set of 9203 training sequences from the X chromosome.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	148	0.889	0.997
2L	23	90	117	0.923	0.998
2R	21	92	166	0.816	0.994
3L	25	92	142	0.915	0.998
3R	28	85	132	0.952	0.999

A set of sequences not annotated as either TEs or exons was generated for the "other" category. All these sequences were taken from the *Drosophila* X chromosome, meaning it was similar information as that used in the first two training sets, just presented in a different form. 9203 training examples were used: 3873 LTR retrotransposons, 2447 exons, and 2883 other. This model had a much lower training error (0.4%) and better performance with an average sensitivity of 89.9% (Table 7.9). This method was tried with a smaller set of examples (243: 112 LTR retrotransposons, 53 exons, and 78 other). Again there was a higher training error (8%) than with the smaller set, but this time the results were better. The average sensitivity went from 89.9% to 94.3% (see Table 7.10). It is interesting to note that with all these training sets, even though the training examples were taken from the X chromosome annotations, performance on the X chromosome was not better than on the other chromosomes.

As many genomes do not have even this level of annotation, another training set was generated substituting consensus sequences from RepBase [65] for the chromosome X LTR retrotransposon examples. This training set used 9186 sequences and had an OOB

Table 7.10: Results for *Drosophila melanogaster* using a set of 243 training sequences from the X chromosome.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	179	0.933	0.998
2L	23	90	123	0.933	0.998
2R	21	92	147	0.934	0.998
3L	25	92	151	0.939	0.998
3R	28	85	103	0.976	0.999

Table 7.11: Results for *Drosophila melanogaster* using training data from RepBase and annotations of the X chromosome.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	162	0.928	0.998
2L	23	90	130	0.940	0.998
2R	21	92	152	0.932	0.998
3L	25	92	154	0.908	0.997
3R	28	85	105	0.972	0.999

Table 7.12: Results for *Drosophila melanogaster* using training data for Eukaryotic LTR retrotransposons and endogenous retroviruses, human exons, and sequences from the Drosophila genome that are neither exons or LTR retrotransposons.

chrom.	size(Mbp)	# annotated	# found	sensitivity	specificity
X	22	96	168	0.922	0.998
2L	23	90	158	0.903	0.997
2R	21	92	190	0.848	0.995
3L	25	92	146	0.915	0.998
3R	28	85	152	0.960	0.999

error rate of 4%. The results are shown in Table 7.11. Comparison of Tables 7.7 and 7.11 shows that using the RepBase files yields somewhat better results (average sensitivity of 93.6%) and also tags more sequences. Tagging more sequences could be either an advantage or a disadvantage, depending on whether the goal is to discover new LTR retrotransposons or to avoid excessive false positives. Most (95%) of the sequences tagged with the first training set were also tagged by the RepBase training set. A disadvantage of using the RepBase training set is that it takes longer to scan the genome. The slow part of the algorithm is looking for LTRs once a putative sequence is found. Using the original training set, the scanner finds LTRs for about 60% of the sequences it identifies; using the RepBase training set, LTRs are found for only about 10% of the putative sequences.

In order to test how species-specific the training set needs to be, a training set based on all consensus sequences in RepBase for eukaryotic LTR retrotransposons and eukaryotic ERVs was used, together with the set of human exons in CCDS[11] [52, 53] and a set of sequences from the *Drosophila* genome that are neither LTR retrotransposons or exons. CCDS is a database of human and mouse genome annotations that is based on a consensus from information gathered by the European Bioinformatics Institute, NCBI, the Wellcome Trust Sanger Institute, and the University of California at Santa Cruz.

This training set had 6758 training sequences and an OOB error of 10%. The results (Table 7.12) were slightly worse (average sensitivity of 91.0%) than the results using

[11] `http://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi`

only RepBase sequences for *Drosophila* and *Drosophila* exon sequences, but better than the results using the training data scanned from the X chromosome.

7.3.1 Analysis Of Errors

One of the goals of LTRsieve is to discover new LTR retrotransposons. Documenting new discoveries requires the help of an experimental biologist and is beyond the scope of this thesis. The purpose now is to establish that LTRsieve has the potential to discover new LTR retrotransposons. To do this the false positives and false negatives found on chromosome 2L of Drosophila using the Repbase training set are analyzed. "False" positives have the potential to be new discoveries. Understanding false negatives helps to identify the weaknesses of LTRsieve.

LTRsieve had three false negatives using the RepBase training set on Dmel 2L. These represent annotations of LTR retrotransposons that were not identified. Of these, two were LTR retrotransposons with other LTR retrotransposons nested inside them. The other was an LTR retrotransposon in the intron of a gene. They were three different types of LTR retrotransposon: 297, Quasimodo, and Max. Future work will address the problem of identifying nests and clusters of LTR retrotransposons. This is a particularly common phenomenon in *Drosophila*; less common in Homo sapiens.

There were 45 "false" positives in this scan. They fall into five categories: those in intergenic regions with no annotations, those in genes in regions containing coding

segments, those in genes in regions with no coding segments, those in clusters or nests of SINEs and LINEs, and those which have been annotated as LTR retrotransposons since the data was assembled.

Fourteen (31%) are in intergenic regions with no annotations. Seven (16%) are in non-coding regions of genes with no annotated function. These are potential discoveries as they are not identified as something else. Further investigation is needed to determine if they are indeed LTR retrotransposons, such as analysis of their structure and coding regions. Five of the forty-five (11%) are actual LTR retrotransposons that were not annotated when data was assembled for testing, but have been annotated since. These represent discoveries relative to the system being tested. So, 58% of the false positives are either potential or actual discoveries.

Fifteen of the forty-five (33%) were in portions of genes with coding regions. These are unlikely to be LTR retrotransposons. It is possible that these coding segments have some relationship to LTR retrotransposons. This is an area for future study, either to determine if these sequences have biological interest or to figure out how to avoid targeting them. The remaining five false positives (11%) were in regions containing clusters of SINEs and LINEs. The system misidentifies these as LTR retrotransposons, because SINEs are of similar length to LTRs and, in these clusters/nests, spaced similarly. This problem motivated the development of the sLTR/SINE classifier discussed in Chapter 6.

7.4 *Homo sapiens*

The LTR retrotransposons in *Homo sapiens* differ from those in *Drosophila* in the following ways. Most of them fall in the category of ERVs. This means they have *env* genes as well as *gag* and *pol* genes. HERVs are thought to be inactive in the sense that they are not reinserting in the genome (although they are active in the sense that they are transcribed). This means that they are mutated and that many are partial. Some of the mutations involve insertions, creating gaps as large as 10K inside the HERV. The human genome has a large number of SINEs (about 11% of the genome) which are easily mistaken for LTRs by the program.

There are no complete annotations of HERVs. Two benchmarks for the results were used: RepeatMasker and RetroTector. RepeatMasker identifies about 4% of the human genome as ERVs. It tags any sequence with homology to a HERV sequence in RepBase. Some of the sequences it identifies are quite short. It often happens that a single HERV is identified by RepeatMasker in several pieces. No attempt is made to verify that an identified fragment is part of a HERV, and, in fact, it is quite likely that many are misidentified. RepeatMasker misses HERVs that have been mutated. It also misses any HERV that does not have a consensus sequence in RepBase. RetroTector uses motif search to identify intact HERVs. It makes no claim to be complete, identifying about half as many base pairs as RepeatMasker, but the HERVs it annotates are well documented. The sets

of HERVs identified by RepeatMasker and RetroTector are overlapping but different: for chromosome 19, 25% of the base pairs identified by RepeatMasker are also identified by RetroTector, and 47% of the base pairs identified by RetroTector are also identified by RepeatMasker; for chromosome 21, 11% of the base pairs identified by RepeatMasker are also identified by RetroTector, and 41% of the base pairs identified by RetroTector are also identified by RepeatMasker.

7.4.1 Results

Since a much smaller portion (about 1%) of the human genome is protein coding, a model based on two categories is used rather than the three categories used with *Drosophila*. The model used has 1334 samples: 774 sequences of length 4800 that are not annotated as either exons or HERVs, chosen at random from the human genome, and 560 sequences of length 4800 taken from the consensus sequence eukaryotic ERVs in RepBase. The random forest classifier generated from this training data had an OOB error of 8%.

When a scan of chromosome 21 is done identifying potential HERVs, about 1% of the genome is flagged as containing HERVs. Since, unlike *Drosophila*, HERV annotations are unreliable, it is impossible to calculate sensitivity and specificity statistics. The best one can do is compare to results of other HERV finding algorithms. Figures 7.1 and 7.2 show approximate proportional Venn diagrams of this comparison. This analysis is done on a sequence by sequence basis. A sequence is considered to be in the intersection

of the sets if there is overlap in the identification. The boundaries are not required to be identical. Two assessments are done because, although RetroTector, like LTRsieve, attempts to identify complete HERVs, RepeatMasker does not. Sequences identified by RepeatMasker include many partial HERVs. Therefore, one comparison is done with all sequences identified by RepeatMasker and another with only those long enough to possibly be complete HERVs. Notice that, in the comparison with all RepeatMasker HERVs (Figure 7.1), the RepeatMasker group is by far the largest of the three: 4538 sequences, while RetroTector identifies 54 sequences and LTRsieve identifies 405 sequences. In the other comparison (Figure 7.2), the RepeatMasker group is the smallest with only 37 sequences.

It is clear from the figures that these algorithms are complementary, each identifying many sequences not found by the others. The fact that the identified HERV sets are also overlapping is suggestive that all the algorithms are also effective. RetroTector is the pickiest of the three algorithms, requiring verification by several different motif methods for each HERV it identifies. Thus, it is not surprising that almost all the sequences identified by RetroTector (89%) are also identified by the other two methods in Figure 7.1. Two of the sequences identified by RetroTector (3.7%) are also identified by LTRsieve but not by RepeatMasker. LTRsieve identifies more HERVs than RetroTector does: 64% of its sequences are not identified by either RetroTector or RepeatMasker. These are either false positives or new discoveries. Future work, in collaboration with biologists, will

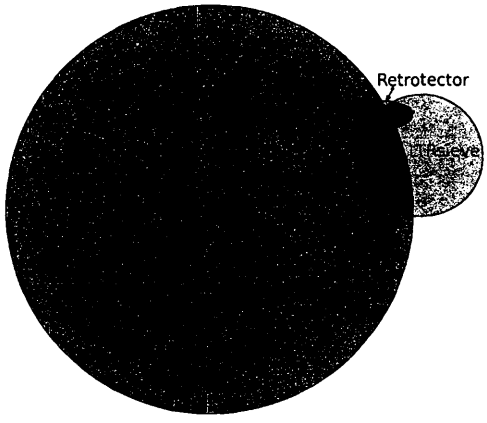


Figure 7.1: Comparison of which ERV sequences were identified by RetroTector, LTRsieve, and RepeatMasker. All sequences identified by RepeatMasker are included, irrespective of length. Note that its sequences constitute a much larger group than those identified by the other two programs.

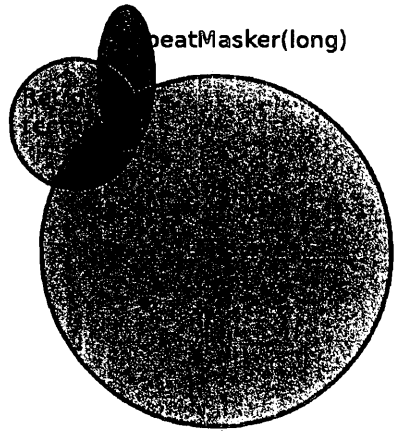


Figure 7.2: Comparison of which ERV sequences were identified by RetroTector, LTRsieve, and RepeatMasker. Only the sequences of length greater than 2880 identified by RepeatMasker are included.

be needed to determine which they are. Figure 7.2 shows that a much greater proportion of the long sequences identified by RepeatMasker correspond to sequences identified by both RetroTector and LTRsieve (54% as opposed to 11%). This makes sense as many of the short sequences identified by RepeatMasker are likely either false positives or fragmentary HERVs. The fact that this is so, however, gives some support to the validity of the RetroTector and LTRsieve identifications.

7.4.2 Conclusion

LTRsieve was originally intended to be a general purpose tool for any genome. This is a challenging goal and there is more work to be done. Some differences can be compensated for with the use of parameters. For example, genomes of different species vary significantly in the number and character of their LTR retrotransposons. Plant genomes can be more than 60% LTR retrotransposons. Human genomes are about 8% ERVs. Fruit fly genomes are about 2% LTR retrotransposons. A parameter controlling the cutoff value for putative sequence identification could vary based on this knowledge. Recently inserted LTR retrotransposons are more easily detected than older ones, having preserved their original structure. More ancient LTR retrotransposons can be badly mutated, sometimes with many insertions and deletions. Knowledge of the age of LTR retrotransposons in the organism might also profitably be used to adjust the cutoff parameter. Species also vary in the gene content of their genomes. For example, *Homo sapiens* genomes are only about 2% genes; *Drosophila* genomes 50%. This affects the distribution of LTR retrotransposons in these genomes: Drosophila LTR retrotransposons are equally likely to be found in intergenic regions and inside genes in introns, HERVs are much more likely to be found in intergenic regions. This knowledge should drive the decision of whether a 2-way or 3-way random forest classifier is used.

An important goal of future work is to modify LTRsieve to deal with the two prob-

lems identified in the tests on these two genomes: the problem of clusters or nests of LTR retrotransposons and the problem of confusion of SINEs and LTRs. In organisms, like *Drosophila*, with known nests and clusters, the check for matching LTRs should be modified to accommodate the possibility. For organisms, like *Homo sapiens*, that are known to have many SINEs, the algorithm should incorporate the SEM feature based classifier developed in Chapter 6 to verify that identified LTRs are actually LTRs and not SINEs.

There is also the issue of training data for the random forest classifier. The results of the experiments using different training sets with *Drosophila* are encouraging. They suggest that it should be possible to construct general purpose training sets that will work well on diverse organisms. Verification of the success of this awaits work in collaboration with a biologist who can verify that the identified sequences are actually LTR retrotransposons.

8 Unsupervised Learning On Tetrahymena IESs

Previously only about a dozen IESs have been studied in detail. Now that the Tetrahymena MIC genome has been sequenced, it is possible to study all the IESs, previously estimated to be about 6000. Biological collaborators say that descriptions based on bioinformatics will help them to formulate hypotheses and design experiments. To this end, ways to divide Tetrahymena IESs into groups are examined.

It is hypothesized that IESs are remains of transposons that have been degraded by mutation [71]. In another ciliate, Paramecium, the IESs have been shown to be related to the Tc1/mariner transposons. These are DNA transposons that transpose in a cut-and-paste manner using a DNA intermediate. Paramecium IESs are much shorter than Tetrahymena IESs.

Not much is known about what sort of transposons could have been incorporated into the Tetrahymena genome. One family of DNA transposons, Tlr transposons, have been identified in Tetrahymena [147]. Thirty of these Tlr transposons have been identified, meaning that versions of them recognizable through biological experiments constitute less than 1% of the IESs in Tetrahymena. In addition, a family of non-LTR retrotrans-

posons, REP, has been identified [48]. No IESs in Tetrahymena have been found that resemble transposons found in other organisms. BLAST searches against REPBASE yield no matches. It would be valuable to biologists to know more about what sort of transposons invaded the Tetrahymena genome and when they invaded.

The mechanism of IES excision has been studied. One result is that a domesticated transposase from a PiggyBac transposon is involved [36]. Transposase is the protein that transposons use to cut and paste themselves throughout a genome. The domesticated transposase from PiggyBac found in Tetrahymena and other ciliates cuts but does not paste. PiggyBac is a type of DNA transposon that was originally discovered in butterfly genomes and has subsequently been found to be common in the genomes of many insects and other organisms. It is of interest to biologists for many reasons: it is found in a diversity of species; it is useful for genetic engineering; it is important in the evolution of baculovirus, a type of virus that infects insects and that has been widely used as a biopesticide [24]; and it is thought to provide a means of horizontal transmission of genetic elements between species [49]. There is a transposable element in the human genome, called LOOPER, that is related to PiggyBac.

Another question of interest to biologists is whether there are transposable elements in Tetrahymena that are actively transposing. An approach to answering this question is to compare IESs in different strains. This was done in the case of the IES called the R element [59]. It was found that there is a short sequence (597 bp) in the R element that

exists in some strains but not others. This is called the R indel. This sequence could be an active transposon.

With the entire MIC genome sequenced, it is possible to address these biological questions through computational means. Three approaches have been taken. First, a BLAST analysis of the sequences was done to get information about how often different sequences and subsequences are repeated. Then, an analysis was done based on edit distance between sequences to determine which sequences were similar to sequences already known to be important, such as Tlr and PiggyBac. Finally, unsupervised learning on SEM features was done in an attempt to categorize different types of IESs.

8.1 BLAST Analysis

8688 IESs were identified. Of these, 5922 had good sequence quality. Table 8.1 reports some basic statistics about the Tetrahymena genome. The number column for the MAC and MIC nuclei represents the number of chromosomes in the MIC and the number of chromosome-like pseudomolecules in the MAC. The MIC is diploid with two copies of each chromosome. The MAC is polyploid with about 45 copies of each pseudomolecule.

Repeated sequences were sought amongst the set of 5922 IESs with good sequence quality by BLASTing them against themselves ($e < 10^{-30}$). Subsequences were extracted that occurred more than once in the set of 5922 IESs. These ranged in length from 68 bp to 4227 bp. Figure 8.1 shows the distribution of lengths. The vast majority of

Table 8.1: Tetrahymena Genome Statistics

	number	size	% of MAC	% of MIC
MAC	181	103 Mb	100%	65%
MIC	5	158 Mb	153%	100%
genes	26,997	65 Mb	63%	41%
MAC non-genic		38 Mb	37%	24%
identified IESs	8688	16 Mb		10%
other MIC only sequence		39 Mb		25%

repeated sequences are shorter than 600 bp. Copy numbers ranged from 2 to 733. About a third of these have copy number fifty or less. Figure 8.2 shows the distribution of copy numbers for the rest. While copy numbers greater than 450 are rare, copy numbers between 50 and 450 occur for hundreds of sequences. Figure 8.3 shows a scatter plot of copy number and length. It shows that it is common to have a sequence of 400 or 500 bp with copy number around 400. There are some sequences with length greater than 2000 bp that have copy number as high as 400, and the sequences with highest copy number have lengths close to 1000 bp. Transposons create multiple copies of identical sequences in the genome. Figure 8.3 shows a large number of sequences with lengths close to 500 bp that have many copies in the genome. Hence, Figure 8.3 supports the hypothesis that IESs are remnants of transposon insertions.

8.1.1 R Indel

The R indel [59] was the first example found of shared sequence between two IESs. It is found in both the R IES and the B IES, two of the ten IESs that were sequenced before

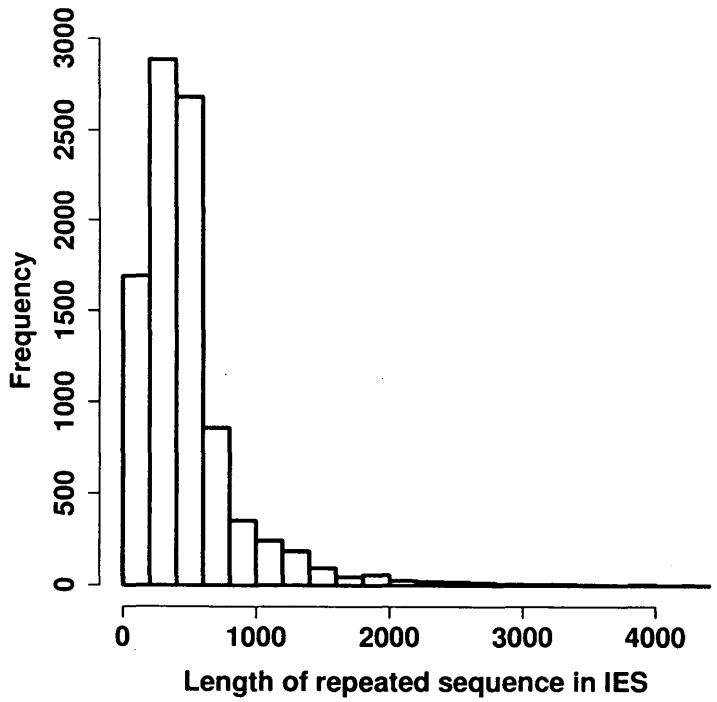


Figure 8.1: Length distribution of repeated sequences within IESs.

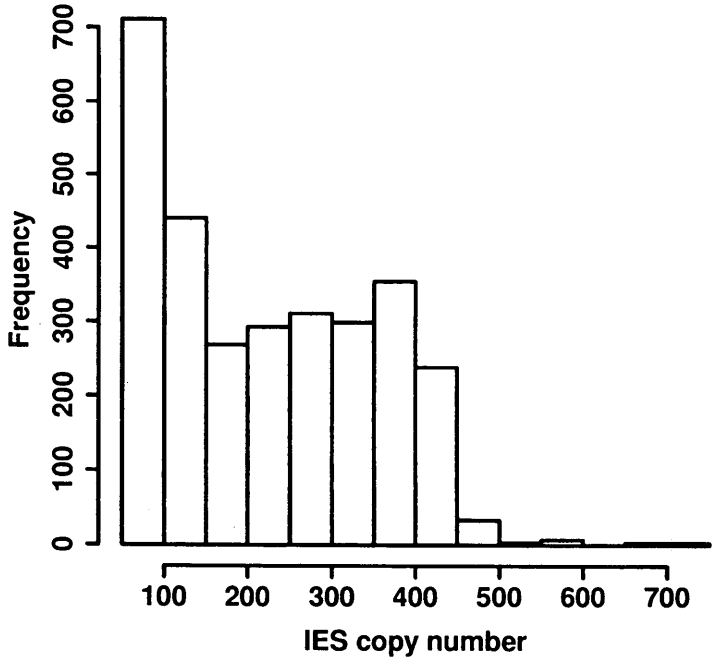


Figure 8.2: Copy number frequencies for sequences within IES with copy number greater than 50.

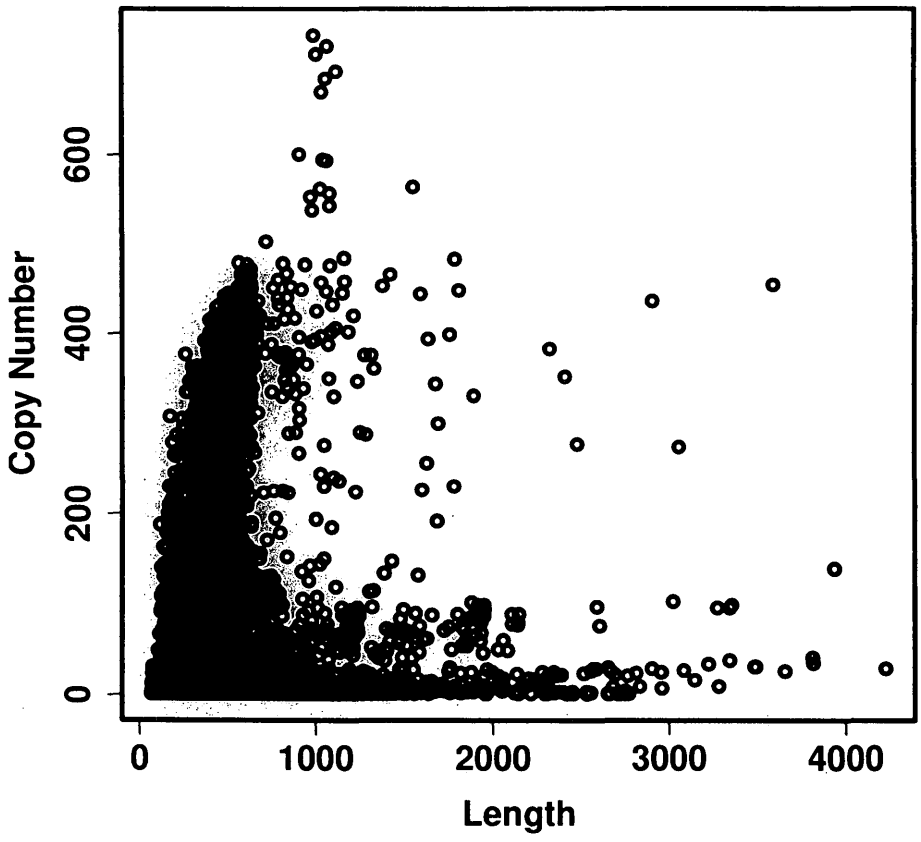


Figure 8.3: Scatter plot of copy number and length of repeated sequences within IESs.

the entire MIC genome was sequenced. It was also found to exist in the R IES in strains B3 and C2 of Tetrahymena thermophila, but not in other strains. Thus, its name. It must have been either inserted in a progenitor of B3 and C2 or deleted from the other strains. 384 copies of the R indel were found in IESs, including two IESs that entirely consist of the R indel. The R indel was also found to exist outside of IESs with 171 copies in the MAC genome.

In addition to the R indel, there are 27 IESs whose entire sequence exists in multiple copies in other IESs, though none with as many copies as the R indel. Copy numbers range from 2 to 58. These IESs are of similar length to the R indel (600 bp).

8.1.2 IESs In Genes

It is known that IESs in Tetrahymena are mostly in intergenic regions. Unlike the IESs in Paramecium, they are not precisely excised. This means there is selective pressure against having them in genes, since imprecise excision could affect the functioning of the genes. The IES locations were checked first against the gene annotations in the Tetrahymena Genome Database (TGD)[12] [135] and then against improved annotations provided by Robert S. Coyne at JCVI [38]. 1294 IESs were identified to be in genes, representing about 22% of the total number of IESs identified. One hundred and three (8%) were in coding regions in the genes. To put these numbers in context, the improved

[12] www.ciliate.org

annotation includes 26,460 genes occupying 63% of the MAC genome. So, IESs are inserted into about 5% of genes, and so only about a third as many are being found in genes as would be expected if they were inserting randomly. Furthermore, 78% of the genes consist of coding regions, so one would expect many more IESs to be in coding regions than were found if there were no selection pressure against it. This means there is selection against IESs in genes and even stronger selection against IESs in coding regions.

Figure 8.4 shows size distributions for all IESs identified, IESs in genes, and IESs in coding regions. All of these categories have the same distribution, but there are fewer really long IESs in genes (more than 8000 bp) and only one really long IES in a coding region. Coding regions have proportionally more short IESs (less than 2000 bp) and a slightly lower median size. The median length for all IESs is 2477 bp; for IESs in genes, 2425 bp; and, for IESs in coding regions, 2279 bp.

Using the improved annotation resulted in a substantial change in which IESs were identified as inside or outside of genes. Forty percent of the IESs found to be within genes using the TGD annotation were excluded using the improved annotation, and thirty percent of the IESs found to be within genes using the improved annotation were not included as in genes under the TGD annotation. This is much more than one would expect given the amount of modification made to the annotation. This suggests that IES insertions complicate gene annotation. It also suggests that the data should be analyzed

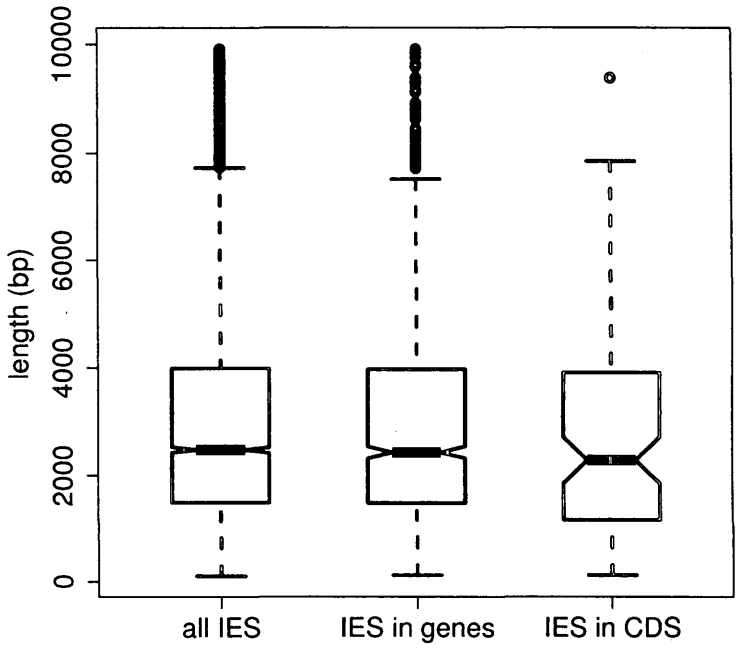


Figure 8.4: Length Distributions for IESs in genes and coding regions (CDS)

with caution, keeping in mind that gene annotation is always a work in progress.

8.1.3 Genes That Have Sequence Matches To IESs

Genes were sought that had sequence homology to multiple IESs, since these could be domesticated genes from transposons. One that seemed to be of particular interest was TTHERM_00934410, described in TGD as a GIY-YIG catalytic domain. The region designated in Figure 8.5 occurs in 28 IESs. It is not known what this gene's function is, but the GIY-YIG domain is associated in other organisms with excinucleases and endonucle-

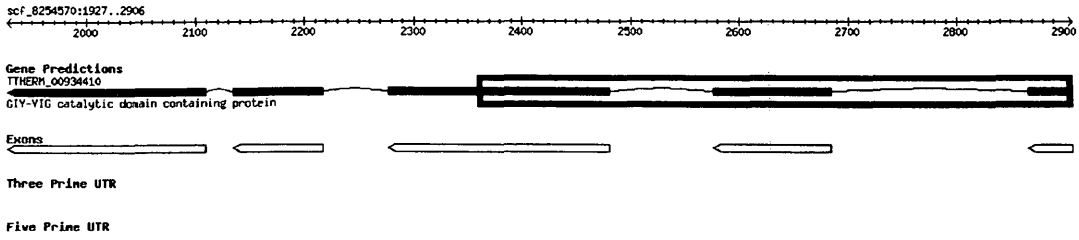


Figure 8.5: TTHERM_00934410 shown in the Genome Browser in TGD with the region duplicated in IESs marked with a rectangle.

ases, including some encoded by group I introns. Endonucleases and excinucleases are enzymes involved in DNA excision, and group I introns are introns that are self-splicing and mobile, like transposons. This makes TTHERM_00934410 a good candidate for further analysis and experimentation.

There are 686 other genes that have sequence homology ($e < 10^{-15}$) with multiple IESs. If a transposon with high copy number was domesticated, the resulting gene would have matches in a large number of IESs. Table 8.2 lists genes with this property. These genes all have sequence homology with more than 70 IESs. Another selection criterion could be the length of the IES match. Table 8.3 lists the twenty genes with the longest average match lengths. These genes are chosen from those with at least four matches to IESs and have match lengths ranging from 47 bp to 549 bp with averages ranging from 196 bp to 443 bp. Notice that the GIY-YIG catalytic domain gene is in this group.

Table 8.2: Genes with sequence homology to more than 70 IESs

gene name	TGD gene description	# of IESs	avg. length
TTHERM_00904060A	hypothetical protein	316	125.25 bp
TTHERM_01120610A	hypothetical protein	250	111.25 bp
TTHERM_01356370A	hypothetical protein	227	92.54 bp
TTHERM_00865150A	WGR domain	216	133.78 bp
TTHERM_00543690A	hypothetical protein	190	96.65 bp
TTHERM_01330050A	hypothetical protein	162	118.37 bp
TTHERM_01211820A	Protein kinase domain	154	81.54 bp
TTHERM_00125670A	hypothetical protein	129	104.69 bp
TTHERM_01141650A	hypothetical protein	121	97.93 bp
TTHERM_00053750A	hypothetical protein	118	81.59 bp
TTHERM_00721920A	hypothetical protein	116	91.66 bp
TTHERM_00954160A	hypothetical protein	116	83.22 bp
TTHERM_00584760A	hypothetical protein	112	78.67 bp
TTHERM_01130800A	hypothetical protein	109	114.06 bp
TTHERM_00935410A	hypothetical protein	80	104.65 bp
TTHERM_00242640A	hypothetical protein	80	62.20 bp
TTHERM_00490780A	hypothetical protein	76	84.00 bp
TTHERM_00399270A	Transmembrane amino acid transporter protein	74	74.96 bp
TTHERM_00681770A	Ras family protein	74	68.35 bp
TTHERM_00188530A	hypothetical protein	72	89.90 bp

Table 8.3: Long IES matches: genes with sequence homology to at least 4 IESs and the longest average lengths for the matches.

gene name	TGD gene description	# of IESs	avg. length
TTHERM_00131170A	hypothetical protein	5	443.40 bp
TTHERM_00298460A	hypothetical protein	5	340.40 bp
TTHERM_00934410A	GIY-YIG catalytic domain	28	303.21 bp
TTHERM_01154660A	hypothetical protein	32	237.50 bp
TTHERM_00197685A	hypothetical protein	33	230.52 bp
TTHERM_00959800A	hypothetical protein	34	224.56 bp
TTHERM_01929220A	hypothetical protein	29	215.24 bp
TTHERM_00737560A	hypothetical protein	30	214.90 bp
TTHERM_01062860A	hypothetical protein	38	213.03 bp
TTHERM_01253450A	hypothetical protein	36	211.61 bp
TTHERM_01350000A	Protein kinase domain	5	208.60 bp
TTHERM_01256620A	hypothetical protein	43	207.16 bp
TTHERM_01471400A	hypothetical protein	30	206.60 bp
TTHERM_01605680A	hypothetical protein	31	201.71 bp
TTHERM_01681240A	hypothetical protein	31	201.71 bp
TTHERM_01382990A	hypothetical protein	45	201.04 bp
TTHERM_01453050A	hypothetical protein	44	200.89 bp
TTHERM_01590530A	hypothetical protein	44	198.66 bp
TTHERM_01222460A	hypothetical protein	46	198.54 bp
TTHERM_01465230A	hypothetical protein	45	195.71 bp

8.2 Edit Distance Analysis

A useful tool for sequence analysis is edit distance. This is a way of measuring how different two sequences are from each other.

Definition 3 Edit distance *is the number of substitutions, insertions, and deletions needed to transform one sequence into another. Edit distance is the same as the global alignment score with scores: mismatch = 1; match = 0; gap = 1; extend gap = 1.*

Definition 4 Alignment length *is the number of bases between the first and last bases of the shorter sequence in the global alignment. This excludes insertions required to equalize the lengths of the two sequences, i.e., insertions before the first base of the shorter sequence or after the last base.*

Definition 5 Normalized edit distance *is the edit distance between two sequences divided by the alignment length.*

To illustrate relationships between IESs in the hope of finding ways to categorize them, multi-dimensional scaling based on edit distance was used. Figures 8.6-8.10 show visualizations of IES data with IESs with various different properties highlighted.

If IESs are evolutionary remnants of transposons, there should be some sequence similarity to transposase genes. All the annotated IESs were BLASTed against a database created from all the transposase genes in NCBI. Figure 8.6 highlights IESs with sequence similarity to those genes. Note that the vast majority of IESs appear to be close to sequences with sequence similarity to transposase genes.

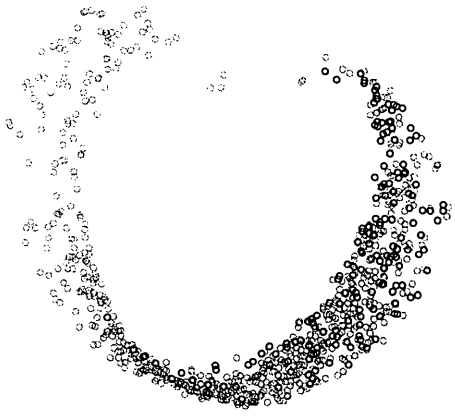


Figure 8.6: Multidimensional scaling of IESs based on normalized edit distance. Highlighted elements are those with sequence similarity (e-value $< 10^{-30}$) to transposase genes.

In [36] it was demonstrated that a homolog of the piggyBac transposase gene produces a protein (Tpb2p) that is likely responsible for the DNA cleavage step of IES deletion. IESs that had sequence similarity to piggyBac genes were sought. Figure 8.7 highlights these. In [147] a family of transposons unique to Tetrahymena, called Tlr elements, is characterized. Figure 8.8 highlights IESs with sequence similarity to these. Note that the piggyBac-like IESs and the Tlr-like seem to occupy different regions of the diagram.

Figure 8.9 highlights the IESs with sequence similarity to the R indel. These are distributed throughout the entire diagram. This is consistent with the notion that the R indel inserts preferentially in IESs as they provide a safe haven. Examples of the R indel

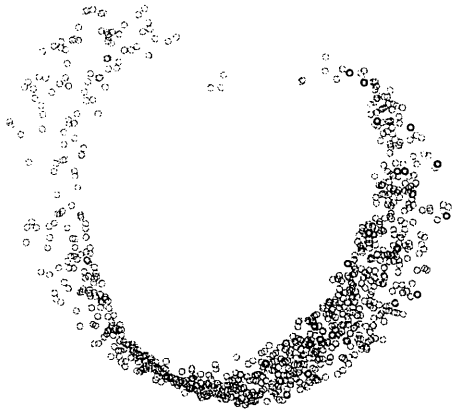


Figure 8.7: Multidimensional scaling of IESs based on normalized edit distance. Highlighted elements are those with sequence similarity (e-value $< 10^{-30}$) to piggyBac genes.

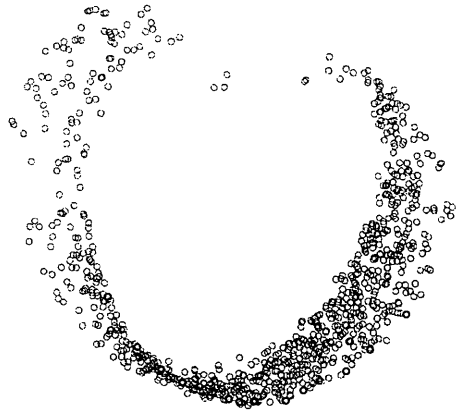


Figure 8.8: Multidimensional scaling of IESs based on normalized edit distance. Highlighted elements are those with sequence similarity (e-value $< 10^{-30}$) to Tlr genes.

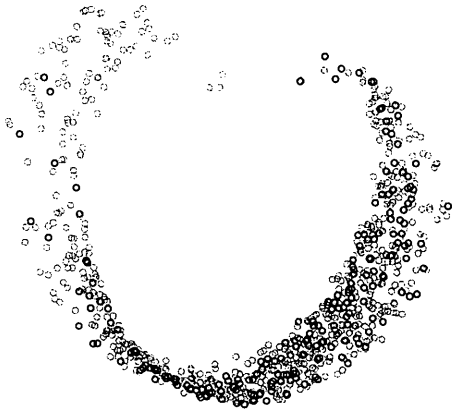


Figure 8.9: Multidimensional scaling of IESs based on normalized edit distance. Highlighted elements are those with sequence similarity (e-value $< 10^{-30}$) to the R indel.

in [59] are all portions of other IESs. Two of the IESs identified are complete R indel matches that are not a part of other IESs.

To identify other sequences that behave like the R indel, sequences that had many copies within other IESs were sought. These are highlighted in Figure 8.10. This group of sequences is in the region of the diagram that does not include any IESs with sequence similarity to transposase genes.

Inspired by the edit distance analysis of sequences similar to the R indel, IESs were divided into two categories: short IESs (like the R indel) and long IESs. Short IESs vary in length from 200-1000 bp, averaging around 500 bp. Long IESs range in length from 1-20K. Short IESs can appear inside other IESs. Short IESs have an average AT content

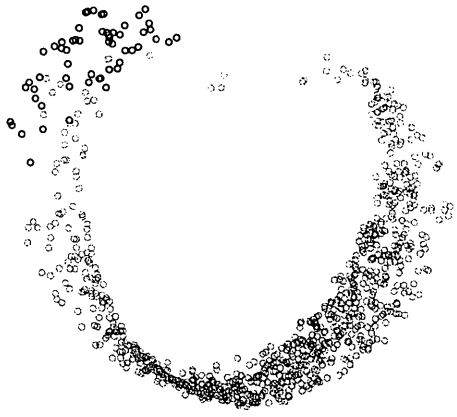


Figure 8.10: Multidimensional scaling of IESs based on normalized edit distance. Highlighted elements are those with multiple copies of their complete sequence in other IESs. This group includes two copies of the R indel. These are the IESs are referred to as short IESs.

of 83%, while long IESs have an average AT content of 80%. Short IESs tend to have higher copy number than long IESs. IES copy numbers vary from 1-733. Short IESs have a median copy number of 18, while long IESs have a median copy number of 11. For long IESs, copy number is based on any portion of the IES that exists in multiple copies ($e < 10^{-15}$); for short IESs only complete copies are counted.

A hypothesis arising from this analysis is that long IESs are evolutionary remnants of transposons like piggyBac or Tc1/mariner relatives and that short IESs are related to the R indel and are SINE-like, possibly active transposons. The R indel has features in common to SINEs (similar length, no terminal repeats, no ORFs), though it has no

sequence similarity to any of the SINEs catalogued in RepBase. This is an example of the type of hypothesis that can be generated by this sort of bioinformatic analysis to be tested in future work in collaboration with an experimental biologist.

8.3 Unsupervised Learning

Most of this thesis has focused on supervised learning in which training data contains class labels. This is the way that SEM features have been used in past research. These have the potential for use in unsupervised learning in which the categories are unknown, and the SEM features are used to figure out what they are. This section explores that potential.

8.3.1 Clustering IESs

The SEM features that were evolved to distinguish IESs from MDSs are not optimized for distinguishing different types of IESs from each other. Therefore, there is a need for a method of creating features that are. In order to do this, the assumption is made that IESs in genes would be different from IESs that were in intergenic regions far from genes. This is a plausible assumption because IESs are selected against in genes. Study of the excision process of IESs in Tetrahymena demonstrates that they are imperfectly excised, with multiple possible boundaries. Imperfect excision would likely have an impact on gene functionality. It is possible that there are multiple different excision methods for

different types of IESs and that the IESs in genes are precisely excised. If this is the case, IESs in genes would likely have sequence features that distinguished them from other IESs.

Working from this assumption, a data set was created that contained IESs in annotated genes and IESs at least 1K from any annotated genes, and evolved features to distinguish them. 4-state SEMs were used with the random forest fitness function. The evolved SEMs were able to distinguish the sequences with 57% accuracy. This suggests that although two clearly different classes have not been found, the assumption was not entirely incorrect. If the classes had been assigned at random, one would expect close to 50% accuracy. The results suggest only that one set is biased towards a particular type (or types) of IESs and the other class is biased differently. These features, however, are evolved to distinguish different types of IESs as opposed to the SEM features used in Chapters 4 and 5 that were evolved to distinguish IESs from MDSs.

The next step was to cluster the data based on the new features. Three different clustering methods were tried. The first used the *pam* function from the R package *cluster* [87]. *Pam* stands for partitioning around mediods and is a robust form of k-means clustering. This method was applied to a diverse subset of 20 of the evolved features selected using dissimilarity clustering. The features were normalized so that they all had a mean of zero and a standard deviation of one. The *pam* clustering was done based on euclidean distance. The second clustering method also used *pam* and the same diverse

set of 20 features, but it used random forest distance for the clustering. Random forest distance is calculated based on a proximity matrix. The proximity of two data items is the percentage of times they end up in the same terminal node when used as oob data for training a random forest classifier. Random forest distance is 1 minus the proximity. The third method used the *KMeansSparseClustering* function from the R package *sparcl* [146] to create two clusters using the entire set of normalized features. K-means sparse clustering simultaneously does feature selection and clustering by assigning weights to the features (some get a weight of zero) and optimizing the clusters and weights so as to maximize the sum of the between cluster sum of squares for each feature.

Figures 8.11, 8.12, and 8.13 show visualizations of the clusters using 10% of the data. Table 8.4 shows cluster statistics for the different methods. The WB and Dunn statistics measure how compact and well-separated the clusters are. The WB statistic is the ratio of the average within cluster distance and the average between cluster distance, while the Dunn statistic is the ratio of the minimum between cluster distance and the maximum within cluster distance. Thus, lower values of WB are better, and higher values of Dunn are better. The pearson Γ statistic measures the correlation between how close data points are to each other and whether they are in the same cluster. Thus, values close to one are best. Entropy measures how evenly divided the clusters are. Table 8.5 compares the clusterings using the adjusted RAND index. The adjusted RAND index is a variation on the RAND index described in Section 4.2.1. It corrects for the similarity one would

Table 8.4: Cluster Statistics

cluster type	WB	Dunn	pearson Γ	entropy
pam	0.74	0.02	0.31	0.68
random forest	1.00	0.47	0.07	0.44
sparse	0.64	0.06	0.44	0.68

Table 8.5: Adjusted RAND Index

	pam	random forest	sparse
pam	1.00	0.00	0.49
random forest	0.00	1.00	0.12
sparse	0.49	0.12	1.00

expect to occur from random chance and ranges from -1 to 1. Values near zero mean no similarity; values near one mean very similar; values near negative one mean very different.

It is clear from the WB and Dunn statistics and from the visualizations that the clusters are not compact and well-separated. This is not unexpected. Biological collaborators expected to find spectrums rather than distinct clusters. One reason for this is that each IES potentially contains many transposon insertions. If, as is likely, there were different types of insertions, there may well be many IESs that contain both types. For this reason, the Dunn statistic is the least reliable as it can be most easily skewed by a few atypical data points. There is no way of knowing which entropy value is best as there is no way to know how evenly divided the classes found actually are, so that value is merely descriptive. Hence, based on the WB and pearson Γ statistics, the K-means sparse clustering was

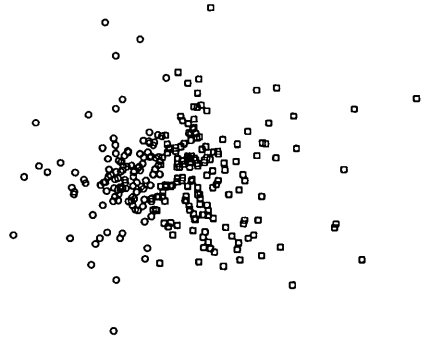


Figure 8.11: Visualization using 10% of the data of the clusters created using *pam* on a diverse set of 20 evolved features and euclidean distance.

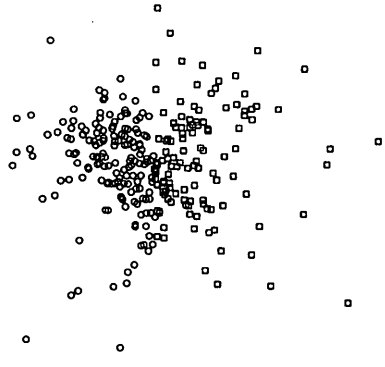


Figure 8.12: Visualization using 10% of the data of the clusters created using KMeansSparseCluster.

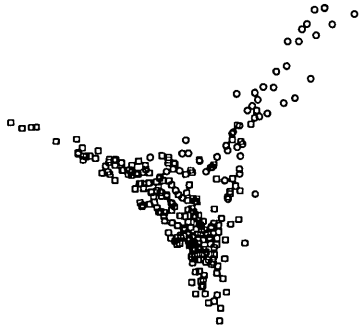


Figure 8.13: Visualization using 10% of the data of the clusters created using *pam* on a diverse set of 20 evolved features and random forest distance.

chosen as the best. The adjusted RAND index indicates that it is not too different from the *pam* clustering. In fact, they agree on 85% of the cluster assignments. Figure 8.14 shows a visualization of the data points on which these clusterings agree. In this figure, the clusters appear to be well separated. This suggests that the disagreements between the clustering methods are mostly on points near the cluster boundary. When calculated on these points only, the WB and pearson Γ statistics are substantially improved (WB = 0.43; pearson $\Gamma = 0.51$), and the Dunn and entropy statistics are unchanged. This is strong evidence that two classes of IESs have been identified.

Once a clustering had been chosen, new SEM features were evolved to distinguish the clusters. This time the SEM features were able to distinguish the classes with 97%

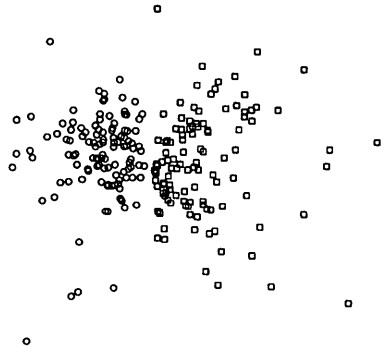


Figure 8.14: Visualization using 10% of the data points on which the k-means cluster method and the k-means sparse cluster method agree.

accuracy. A random forest classifier was also built with k-mer features with $k = 1 \ldots 3$. This classifier was able to distinguish the two classes with 95% accuracy. This suggests that two distinguishable classes of IESs had been found. The next step was to attempt to describe the differences between the two classes in a way that is meaningful to biologists. The process followed is summarized in Figure 8.15.

8.3.2 Cluster Analysis

It is possible that the distinction is not biologically meaningful. It is possible, for example, to distinguish people based on hair colour with high accuracy, but the distinction is not useful if your goal is to determine disease risk. To determine whether the distinction

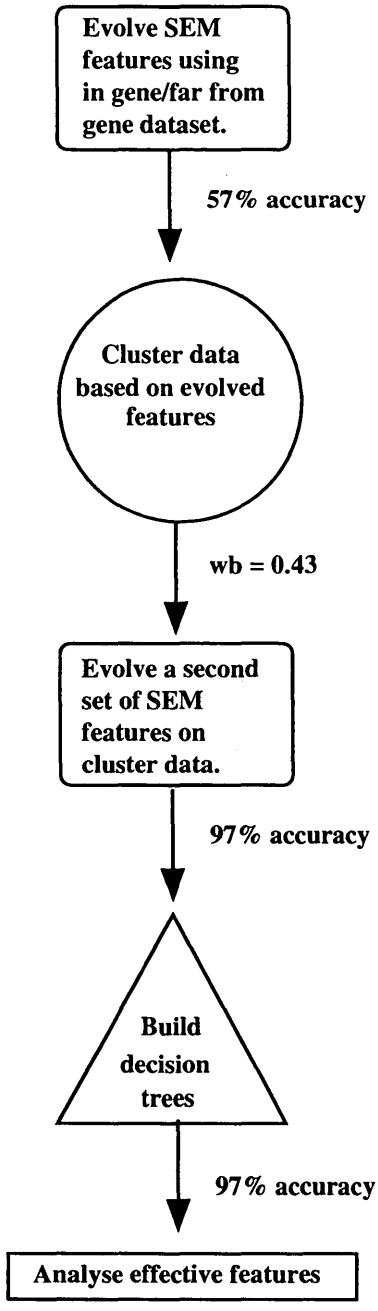


Figure 8.15: Process followed to find descriptive features for two classes of IESs.

is useful, the help of a biologist is needed. However, it is necessary to give the biologist meaningful information with which to work. It is unlikely that even an expert glance at two sets of thousands of sequences will lead to any result. Wet lab experiments on the sequences could lead to insight, but the biologist needs some sort of hypothesis to work with about what is different between the two types of IESs.

Understanding the features that best separate the two clusters could lead to understanding the differences between the sequences. To determine which features these are, decision trees were built to separate the sequences in the two clusters using the features. The decision trees were built using recursive partitioning with the R package *rpart* [130]. Figure 8.16 shows the decision tree built from the SEM features evolved to separate IESs in genes from IESs far from genes; Figure 8.17 shows the decision tree built from the SEM features evolved to separate the two clusters, and Figure 8.18 shows the decision tree built from k-mer features. The SEM features in the trees are named based on their absolute correlation distance to k-mer features. The name is of the form sem_XX_n.nn. The XX refers to the closest k-mer feature. If the correlation of the SEM feature with it is positive, the name of the k-mer feature is given in uppercase; otherwise, in lower case. The number at the end of the name is the absolute correlation distance between the features.

For the tree in Figure 8.16, the classification of 97% or more of the points is determined by the root node of the decision tree. The second level of the tree affects only a

few IESs near the boundary between the two clusters. Furthermore, the features in the trees are correlated with each other. The three features in the tree have Pearson correlations with absolute value greater than 0.70. Thus, the focus is on the SEM feature at the tree root.

In order to have some more SEM features to analyze and because of the knowledge that the evolved features had diverse effective features, two more decision trees with evolved SEM features were built. These decision trees were built excluding all SEM features with high correlation (absolute correlation distance less than 0.2) to the features in the roots of the trees in Figures 8.16 and 8.17. One tree was built using the features from the first evolution excluding all features with absolute correlation distance less to or equal to 0.20 from the feature at the root of the tree in Figure 8.16. The other tree was built using the features from the second evolution with absolute correlation distance less than or equal to the feature at the root of the tree in Figure 8.17.

8.3.2.1 SEM Features

The feature at the root of the decision tree in Figure 8.16 is sem_tt_0.48. The SEM that generates it is shown in Figure 8.21. It is generated by the state in the upper right corner of this SEM. This feature counts Cs and Gs. It counts all Cs except those counted by the state in the upper left. The Cs that are excluded mostly follow Ts (sometimes As). This feature also counts some Gs. The Gs that it includes mostly follow As.

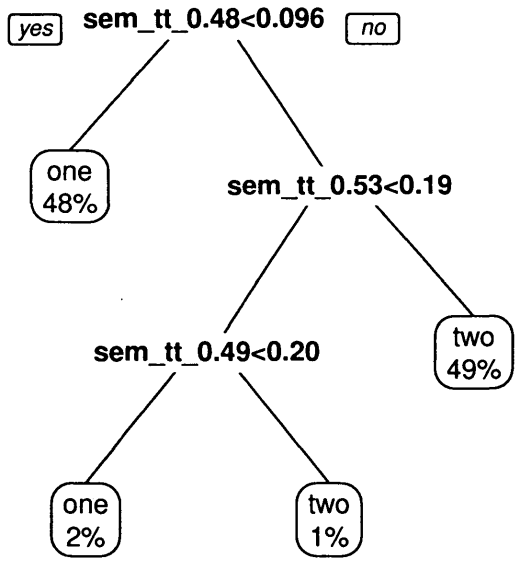


Figure 8.16: Decision tree built using the features generated by evolution distinguishing data sets containing IESs in genes and IESs more than 1K from genes. This decision tree was built to distinguish cluster one from cluster two, reserving 20% of the data for testing. It gets 97% accuracy on the test data.

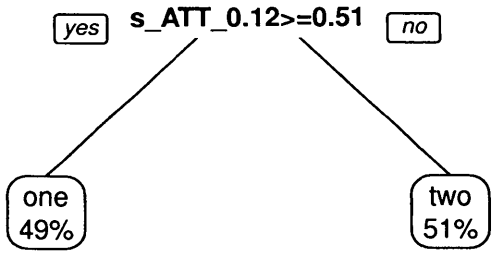


Figure 8.17: Decision tree built using the features generated by evolution distinguishing cluster one from cluster two. This decision tree was built to distinguish cluster one from cluster two, reserving 20% of the data for testing. It gets 97% accuracy on the test data.

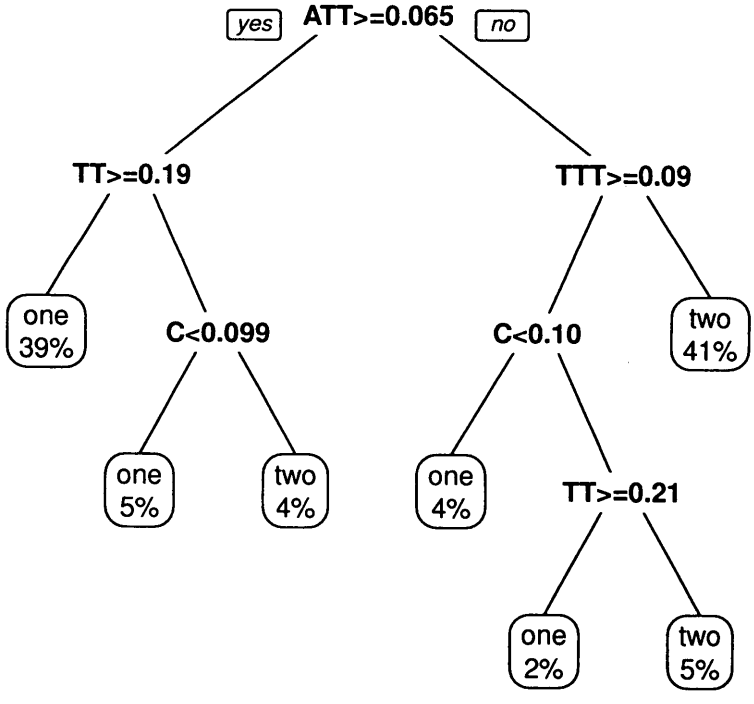


Figure 8.18: Decision tree built using the k-mer features for $k = 1 \ldots 3$. This decision tree was built to distinguish cluster one from cluster two, reserving 20% of the data for testing. It gets 91% accuracy on the test data.

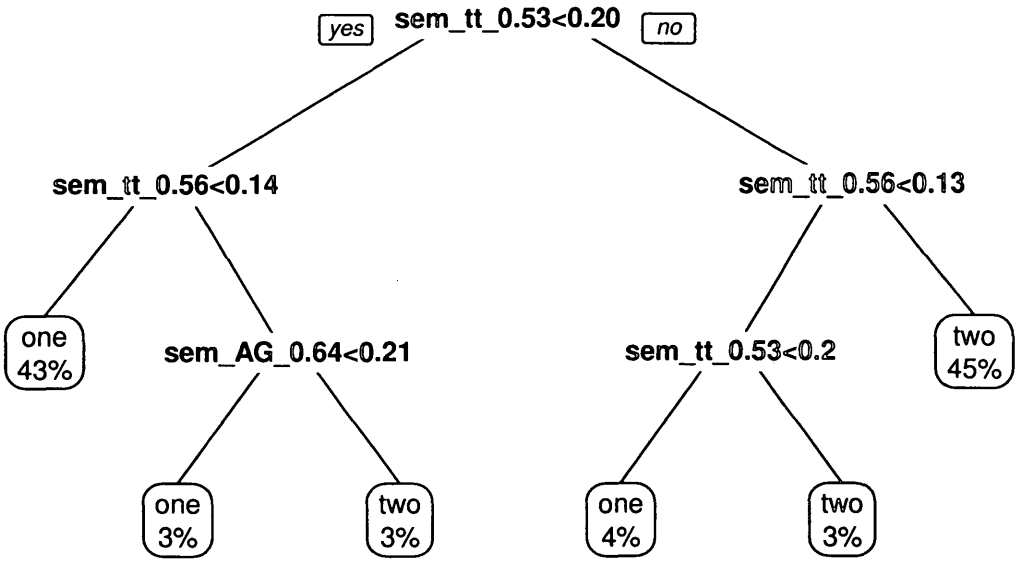


Figure 8.19: Decision tree using SEM features from first evolution excluding all features with absolute correlation distance less than 0.20 from sem_tt_0.48. This tree achieves 97% accuracy on test data.

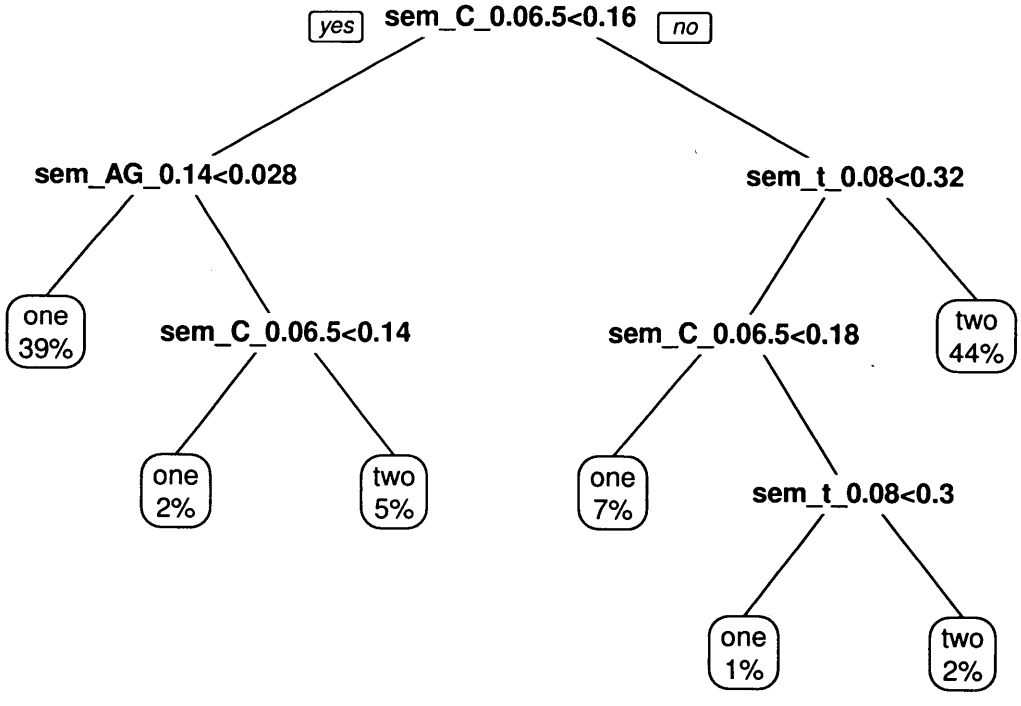


Figure 8.20: Decision tree using SEM features from second evolution excluding all features with absolute correlation distance less than 0.20 from s_ATT_0.12. This tree achieves 93% accuracy on test data.

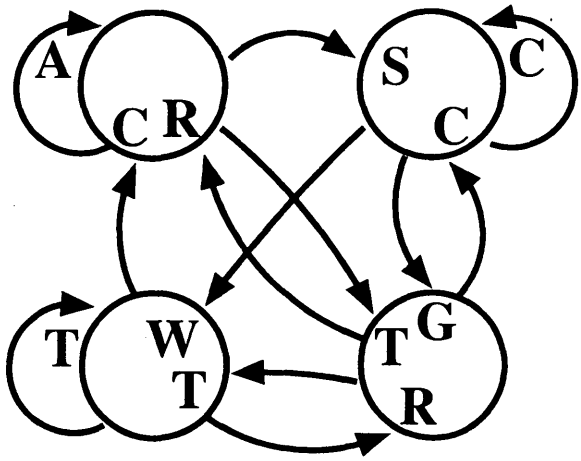


Figure 8.21: SEM at root node of the first tree built from the first evolution (sem_tt_0.48).

The feature in the decision tree in Figure 8.17 is sem_ATT_0.12. The SEM that generates it is shown in Figure 8.22. Its value is counted by the state in the upper right. It counts Ts and As. It counts all the Ts in the sequence except those counted by the state in the lower right. The excluded Ts always follow Cs or Gs. The As that are counted never follow Ts. Thus, this SEM feature is excluding TAs from its count.

The feature at the root of the decision tree in Figure 8.19 is sem_tt_0.53. It is generated by the SEM in Figure 8.23. It is counted by the state in the upper right. The feature counts TAs and non-T bases that follow $TAGT^*$ or TGT^* or HST^*. In this case, T^* means some number of Ts (could be zero).

The feature at the root of the decision tree in Figure 8.20 is sem_C_0.06. It is calculated by the SEM in Figure 8.24 using the state in the upper left. This feature counts all Cs except those counted by the state in the upper right. Those excluded Cs always follow

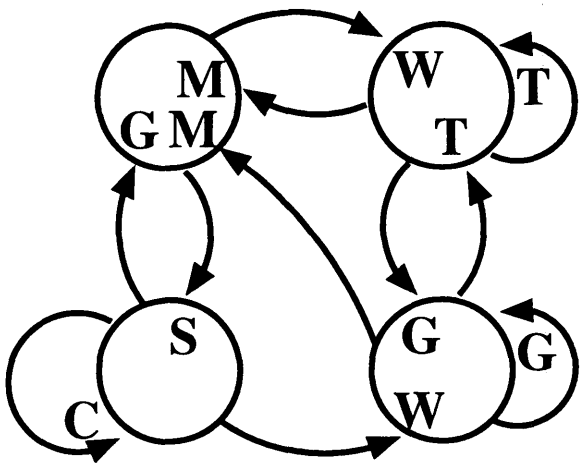


Figure 8.22: SEM at root node of the first tree built from the second evolution (sem_ATT_0.12).

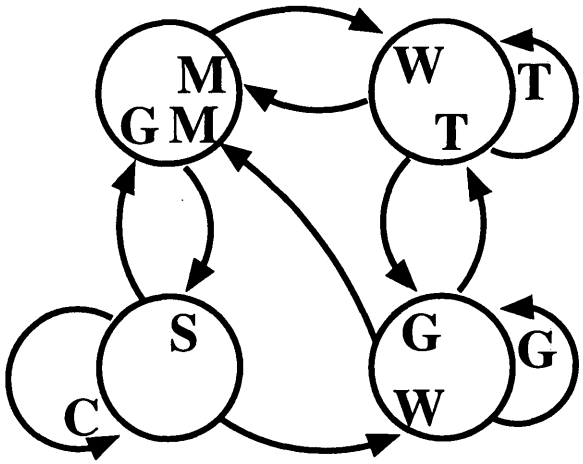


Figure 8.23: SEM at root node of the second tree built from the first evolution (sem_tt_0.53).

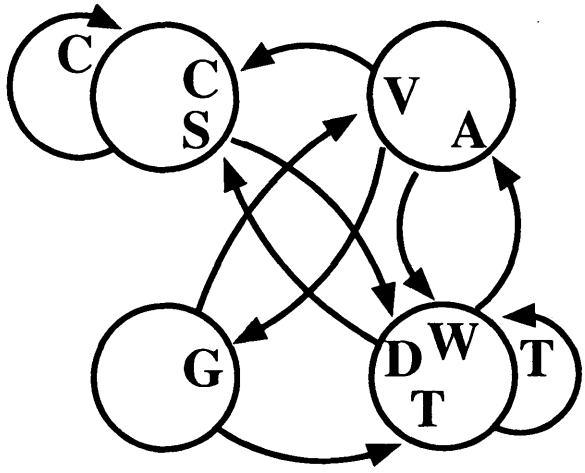


Figure 8.24: SEM at root node of the second tree built from the second evolution (sem_C_0.06).

Gs. So, some GCs are excluded. There are also some Gs included in the count. These often follow Ts, but never Cs. So, TGs are included, but not CGs.

8.3.2.2 Best K-mer Features

The tree built using k-mer features in Figure 8.18 achieves 91% accuracy distinguishing cluster one from cluster two, while the trees in Figures 8.16 and 8.17 built using SEM features both achieve 97% accuracy. This suggests that, although the differences between the two classes can be described using k-mer features, the added descriptive ability of SEM features yields a meaningful improvement. Note that the k-mers selected are close to the SEM features selected. ATT is at the root of the k-mer tree, and a SEM feature close to it is at the root of the tree in Figure 8.17; a feature close to TT appears at the root of the trees in Figures 8.16 and 8.19, and TT appears in the second level of the k-mer tree; and a feature close to C appears in the root of SEM2notV78 and in the third level

of the k-mer tree.

8.3.3 Representative Sequences

Based on the *pam* clustering using the best features in the second evolution, the medoids were selected as representative sequences of the two classes. These sequences are close to the same length, around 3K. The representative of class one (sequence shown in Figure 8.25) is found in an intron of TTHERM_01081810. This gene has a homolog in Paramecium (GSPATP00001695001). It is described in TGD as a "chlamydial polymorphic outer membrane protein repeat containing protein." This suggests a similarity to a gene in the bacteria *Chlamydia* that is believed to provide protection to the bacteria from the immune system of its host. This IES has a subsequence of length 312 bp near its beginning that matches subsequences in two other IESs and a subsequence of length 364 bp near its end that matches subsequences in three other IESs. The representative sequence of class two is shown in Figure 8.26. It is a distance of 1788 bp from any known genes. It is found on supercontig2.126:161617..165045 in the MIC genome, and is deleted from between TTHERM_00527150 and TTHERM_00527160 in the MAC genome on scaffold 8253811:387757. This sequence has a 366 bp subsequence that matches subsequences in 106 other IESs, a 1329 bp subsequence that matches subsequences in 143 other IESs, and a 534 bp subsequence that matches a subsequence in one other IES.

Figures 8.27 and 8.28 show the bases that are counted to calculate sem_tt_0.48 for

AATTTCTAGATAATATTGTAAAAATTTTTTAAACTTTTATTAAAAATTCTTCCAAAATAAAGCACTTCAATGTTCTAAATTATTATGGAA
GCAGAAATAAAAAAATTTAATTCTTTAATTTGCTATAAAGTAGATTAAATAAATAAATAATTTTATTTTAAAATCAATGCCTTATGATTT
TTAATTTGAGTATGAAATTTCAAATAGTAAATTTAATATTCATAATAATTTTTTTATTTTATGATAAAATATGCTTAATCCTTTGAAATA
AAGTATCAATAATTTGAAAAAATAAATAACTACCAAATTATATTAGCAAATAAATAATTATATAATTTGGATTATGATGTAATGGAGCTT
TTTAAAGTAAAGACCTTCATGCAAAAATAAAATTATTCTGATTATTAGTTAATATAGGGATCCTATTAAAATTTTAAGTTCATGAAAGGA
AGTGATCTTCCATATTTTTAATCTTTTATCGCAATAATATTATATCAATTGATCAAGATAATAGAAGTCATATTTTTCTCAAGAAAAAAA
GTATGTTTTCACATTTCATTCTCATCTTAATGAAAATCTTAAAAAATTAATAAATAGAAAATTCTACCCCACTATGAGAATTACAAGTAA
ATCCTGTTGTCCTAAATCAAAATTCAGGATTTGGAAACTAAAATTTTTTTAAAAAGCTAAAATAAATAAAAAATTGAATGTATTCAATTA
ATTTAAAATTAATAATATTATAATTCAAAATCTATTATAAATATTTTCATGCTTTAAAAAATTATTTAAAGGTAAATTAGTGATGCCTCT
TCACTAGATTTTAGAAAGGCATTTAGTTATATTTAATTGATTTAAAATTCATAAAATTCTATTATTAAAATTTTTAAAATAGAAAAATAT
GATTTATAAATAAACCAAAAATAGTAATGCCATATTATTAGATTTTTTTAAGCATTTTACATTACAAAAATTCAAAAATAAAATTTTTGA
TTCAGTATACACAAAAATTGATTTTCTGGAAGTGCATATTAAAAACCATAAAAATTATAAAGCTAAAAAATTATTAATTACAACTAAATA
CTTGCTTATTAATAAATGATATATGTTTTAGTTTAGCTTATATATACTAATGTAATATTTCAGTCAGTAATTAGAGATTAATACAAAATT
TAGTTTATAGACACTTTAGCCATATTTTATTTATGTTTACTTAATTCATTTAAAAATTTATTAACACATATCTTTAATTGTTCAAACTTT
AAATATAAATAAAATGCTAAAATATATTTTCTCAATTTTAGAAATTTTCTGTAAGTATTTAGGATATATGAAATTATATACATCTAATTG
TGATTATTTATTTTTTTTCTTTTATCCATTGAATTTTAATATTACTCTGAATAATTTCAGTTATTTAGTTAGAAAATTTTGAGATATTTT
TTTCAAAATTATAAGCTAATATATACTCATATTTTTAATTTTAATGTTCAGTATTATTCTAATTTTATTGATTGTTTTCCATAAATTTTA
TTTTGAGATAAGTGGTCATAAAATATTAAATGAAAAAAATTTTCAAAAAAGTAAAATTTTGTATTAGTCTTCATAAAATATTATATTAGT
TTCTATAAATTATACCCTAACCAAAGTGTAAAAATTTTCATTTGAAATAAAAAATAAATTTTTTATAAATATTTTTAAATGATATTATAT
AAGTAAGGGATAAAGCAACTAATTCCCCACCCATGCAAAATTAAATCATATTTTTAAAATTTATATTAAATTGGTTATTATTATCTACGA
AATAAATTTTATTTTATTAATTTTAAAATTCAAAAAAAAGGTTTATAAAAATATTTTTAAAAATAATATTTTTTTTATTTAGTTAATATA
AGATAAAAACCAATAATAAATACGATTTAAAAAAATATAAATAGCATGATTTTAACATTTTAAATAATAAATAATCAATCAATGAATTAG
TTTAAAAGCCTCAATCATAAATAAATTTGACATTTATTGAATGATAAAAAAATATTTTTAATAAAAAAATAAGAATAGTTTTTTAAACTA
TAATCCTTATCTAACAATAATTATCATATTAATAAATATATAAAATACCATAAATCCATCATCGTATATTAGAGCTCGATTATATTAAAA
CCAAAAATTTATTTTGTTTTTATTTAGATAACATTAATACCCTTTTTAAAAAATACTTATAAAATTCCATAACAAATTTATAAAATTTGA
AATAAATAAATAATTTTAAATCTTATATTGTGATTTAAATTATTTTTTAATTATATATTTTCAATTTAAAATAATTTAACAAAAAACTTC
ATTCATTATTTAATATTAAGAAAGAAGAGAAAGGGAAGAGTTAGTGTTAATGTATGAGGATTTAATATACCTTAAAAGAAAACTATGATT
AATTAAATTTTTTGAATAAATTGAATATACTTATACTCCTCACATTTTTTACAAGCCCTACCATTTATGTTTAACGAATAATATATGTAG
AAATAAAAATAAATTATGGATTAAAGATATTCTGGAATTATTCTATGAATCAAAAAAATAAAATAGGGAGCTGGCCTTTTTTTGATTGAT
TAAGAATAAAAATTTAGAGATTAAATTAAAAGGAAGGTATTTGTTTCTTTTAAAATTTTGCAAAATATTAAAATTATTTGATATTACATT
CATTTTCAGCTAGAAAATACAATGAATTTGAAATTTTTATCAAAATTTGAAAATAAATAAATACCTTCCTTTTAATTTAATTTCTTTATT
TTTATATTTTTAATTAATAGAAAAATATCAGATGTTTTTAATGTATTATTGTATAGAAAAATTATTGGAAAAAATGAATTGAAAAATATC
TACCTTCAGAAAATAACTTAAAATATTTTCTGCAAGGAAATTAATTTGGTAGAAAAAAAACTCAAAAATAAT

Figure 8.25: Representative sequence for Class 1. This sequence is 2952 bp long.

the two representative sequences. This feature has a negative correlation with the k-mer feature TT. As expected, no TTs are counted. It is smaller for sequences in class one than for those in class two. Note that among the bases counted are runs of Cs and instances of the dinucleotides CC and GC.

Figures 8.29 and 8.30 show the bases counted to calculate the feature s_ATT_0.12, which was determined to be the most effective feature resulting from the second evolution of SEMs. This feature has a small absolute correlation distance from the k-mer feature

ATT, and, as expected, many ATTs are counted in the sequences. Also counted are many runs of Ts and instances of the dinucleotide AT. Notably absent is the dinucleotide TA. This feature is larger in class one than in class two.

Figures 8.31 and 8.32 show the bases counted by the SEM feature sem_tt_0.53. This feature, like most from the first SEM evolution, has a negative correlation with the k-mer feature TT. Thus, there are no TTs included in its count. The absolute correlation distance from sem_tt_0.48 is 0.23, meaning that it is not all that different, despite the fact that it is mostly counting As rather than Cs. Its value is lower for class one than for class two.

Figures 8.33 and 8.34 show the bases counted in the representative sequences for SEM feature sem_C_0.06. This feature is highly correlated with C-content. However, note that many Gs are also counted. Its value is lower for class one than for class two. It has absolute correlation distance of 0.21 from s_ATT_0.12.

8.3.4 Conclusion

This chapter demonstrated how bioinformatic analysis and, in particular, bioinformatic analysis of SEM features can be used to provide information for biologists which can inspire them to develop hypotheses that can be tested experimentally. There is much more work to be done describing the *Tetrahymena* genome bioinformatically. This work is best done collaboratively with biologists. Several directions suggest themselves immedi-

ately, however. In this work, two types of features have been used to cluster and describe different types of IESs. It could be valuable to explore other types of features. In particular, features that involve finding motifs are likely to be useful. The IES classes created using unsupervised learning were based on features evolved with a somewhat arbitrarily chosen subset of IESs. Repeating the process on different sets of IESs, for example those with small edit distance to transposase, Tlr, or PiggyBac genes or maybe even randomly chosen sets, could lead to different insights. Also, it is possible that IESs have an orientation. Transposons certainly do. If that is the case, then evolution is being performed on sequences with both orientations. Since both k-mer and SEM features are sensitive to orientation, this affects the quality of the features. Finding a way to determine the correct orientation could significantly enhance the results.

Once sets of representative sequences have been found for different types of IESs, consensus sequences can be built for them. Previous researchers have had no luck finding transposon sequences with homology to IESs [59]. This could be because the transposons that inserted into Tetrahymena are not included in Repbase (the database of transposon sequences), perhaps because they are unique to Tetrahymena. However, it could also be because the sequences have been mutated beyond recognition. In this case, it might be possible to find a related sequence in another organism based on a BLAST search with the consensus sequences.

TTTGCAGAAAATCTTATTTTTCGATTTAATAAAAAATTATTTCCTAATTTTGATCGAAAATTATTTTTTTTACATTTTATCCAAAAACGA
ATAAATTTTTTTTAAACTTTTTCGAAAACAAACATTCATTCTTAGCTCAAATAGAAGGTTTTGCTGTTCATTTATATTTGAAAATAAATA
TGAAGTTAAACCTTCTTTTTGTTTCTATTTTTTCAATTCGAATACTTTAATTAAATTTTAAGAATTTACAGCAAAAATTCAATTTTTTGC
ACCAAAAATATAAATACTCTGCTTGATTTTTAAGTAAAGGTATATCTGTCAGATAAATTCTGATTTAAAAATGAAAAAAAAATACATAAA
CTGCTATTTAGTCATTTATATATTAATATTTTTACAAACTCTGCTGATTATATGTATCCAATGTGCGGAATTTTAAAACCAGTAACTGCT
ATATTAAAAAATTTTCAAAAAAATGATTGTCTTCTTTTTATGAAAGTAGCATTATTCGATTTTATTATAATTTAAAATTTTTTTAATCTT
TAAAATATTTTTAAATATATTTTATACTACTTATAATTAAAAATATCTGGTCAAATTCATAGATTTCGACACCATAACCTAATTATTGTT
TTTTGACATCTGTTCCTAATTCATTTATTTAGACACTGCCCCTCTATATTAATTTTTGACACTAAGACTTAAATTATAGTTATTAACACT
ATATCCAATTATTTAATATCGATACCAGCATCAAAGAGCATGTATCAACACTAATTTCGAATTTTAAAATCCTTCTAATTTTGATGTTTA
TAAGGTTTATCTGAAAGAAATATTTAAATTAAAATAATGAAATAAATTATGAAATTAATTAAAAAAGTATATCTAAAGACAATAGACCCT
ATTTTTAATAATACCAAACTCTATTTATCATATTTAATAAAGCTATTTATTTAATTGCTTCTTTCTTAAATAAATATTGCAACTATTATT
TAAATTATTAATATAAATCATTTATAATTATCTAAACCAAGAAAACATTTATAGAGTGTTATAAAATTTTACGTATAACGATTTGCACTG
ATGAATGGAAGGGTTATAGTAATCTAAAAAAATTATAATATCAACACCACCCAATCATAAAAAAAATTTTGTTGCACCATCAACTAACTT
AGATTAAAAAAAAAAAGAAAGAATTATACTAAATCAAATAGAAATAAACCATAAGAATAAAAAAAAATAGTTTAACCTATTAAAAAAAACA
AAAATTATTTAATGGTCATATATTACATACCCAAGGAATAGAAAATAAATGGAGTCATTTTAAAAATAGTCTCCATCAGCTAAAGGGGTA
AAGAACAGACTTCATTATAGATATTTTAGTTTCATTATTCCTTTTAAGTTTTAGGAATGAAATTGAATAAAAAAGCTTTATAAGCTTAAA
ATTAATATATTAAATAAAAAATAAATAGTAACTTGGAAAATGATGATGAACAGTAAATTTAAGAAAATAAGCAAATTGAAGATTAAGAAG
AAGAAAAAGATCATACTCGTATGAACTCCAAGCTTCTAATCTAATAAATACAAAAATATTCTAGTAAGGATTATATGGCTGTACTGTTAT
AGATAACTTAAACAGGCATATAATACTAAGAATCGTACAAAATTATTAAAAATCATAATATATAGCTGTTCCTACATTTGAAATATCACG
GATCATGATATCATGAATTGTGATAATCTCAAATAGTTTTTATATTACTAATGATTTTTTTTTTTTTCTAAATCTTAATTATTTTCTTTA
TAAAGTGAAATGCCTATTTAATTTACAAAAGGCGAAATGTAATATACCTTAATAACATAATAAATACAGCTAAATAAGCTTAAAACAGCT
TTATTTTTGAGTTTTTAGAAAGCTTTAAATTCGCAAAATGTAAAATTTGAAATAATAGATACCGTATATTCTTATAATATGTCATTTTTT
TGAAAAAAATAAATTTTGAAAATTTTTTCGAACTCAAAAATTAAATTTTTTTAAAGCTTTTTCGAAAACAAACATTCTTAGGTAAAATAG
AAGGTTTTGCAGTTCATATATATTTGAATGAAATATTAACTTAAACCTTCTTTTTCTTTCTATTTTTTCAATTCGAATATTTTAATTAAA
TTTTAAGAATTTACAGCATAAATTAAAATGCTCTGATTGATTTTTAAGTAAAGGTATATCTGATAAATTCTGATTTAAAAAAGAAAAAAA
AATACATAAACTGTCATTTAGTCATTTATATATTAATATTTTTACAAACTCCGCTGATTACTTACATATATCCAATCAGCAGAATTTTAA
AACCAGGAACGGCTATACTTTCAAAAGATGAAGCCTCATTTTTAGAATAACTTTATAATAGTTTATACTTTGAATCATCTAAAGGTAAAC
ACCATTACATGAAAAATGGAGTACAATAAAGCTCCGCTCTCTCCAATTTTATTTAATATCTATATGAATGAAATAATTCTAAAAATTAAA
TAAAAATTTGTCAATGAACTATTTGAATTAATTTTTAATAATCTAAAATCATAAATTAAGATTTTAGGTAAAAGATTACCAAAAAATATT
TTATTTAAATTAAAACTTTGAACTAAGTAACTCAAACTTACAATTACTTAGGAATAAACATAAACAGTAGCAGAAACTTAATGCCTCATT
TTAAATTTTGAAAATAAAAAATAAACTTCCTCTAAAATAGTACTCTATTTTATTTAAAACATTTGCAAATTTATAATAGATTTCTCTTTT
GTATTATATACATTAGCCTAATGTATTGCATGCCATTGCTAAAGGCACTTAGAAAAACCACCAAATCCAAATTTTAAACTTACTTTCCAC
AAAGGAAGCTTTAAAAAACTCATATGTAAATGCCTTCTAATTCTTCTATTAGATATAATTCTGTTTTCACAAGTAGATAAATGGAAATAA
AAGACTGACTTCTAAATTATATACATTTAGAAATTAATAAAACATATTGTTATTTTTGTTTGGTTATTTTACAATATTTATGTTCTTGAA
ATTTCTTTACCCCATACAAATTTATACGTTTAGGTTATTAATAAATTTACATTAACTAAATTTTCACCTAATTATATTACTTCGACACTA
AAGTTAATTTATGATTAGTCAACAACACTAAATTTTGTTGTGTCAATAACTATTATTTTAGCCATAGTGCCGAAAATTAATAATAGCGGG
ATAGTGTCAAAATATATAAATTGGGAACAGATGTCGAAAACCATTAATAAGGTTCTGGTGATGAAATCTATGAAGTTGACCAAATATCTA
TATTTAATATTTATTTTTGCCTTATTATTCGGTATATCACCTAGCTTTTTTTAAGTTTGCTGAATTTGAATTTTTTAAATTTTAAACAAA
CCAAATATA

Figure 8.26: Representative sequence for Class 2. This sequence is 3429 bp long.

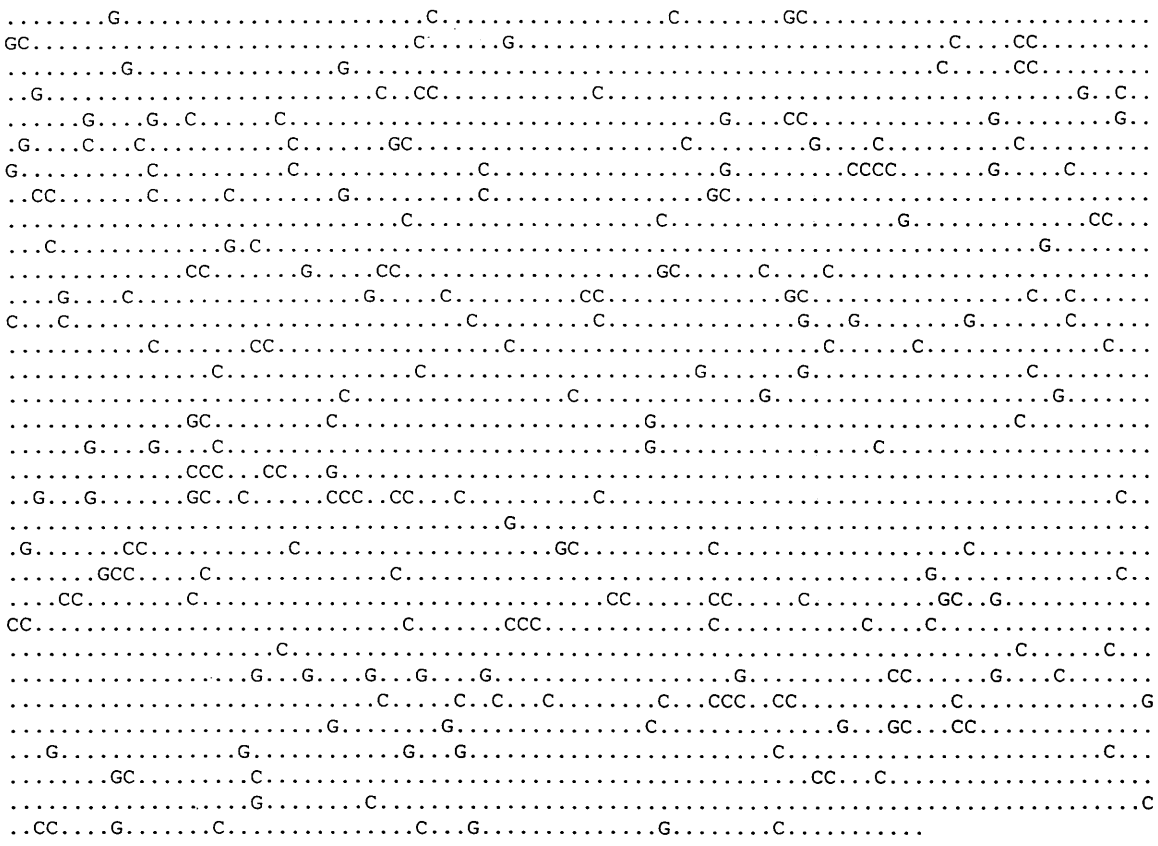


Figure 8.27: Bases counted for feature sem_tt_0.48 for sequence representative of Class 1. These represent 8.0% of the sequence.

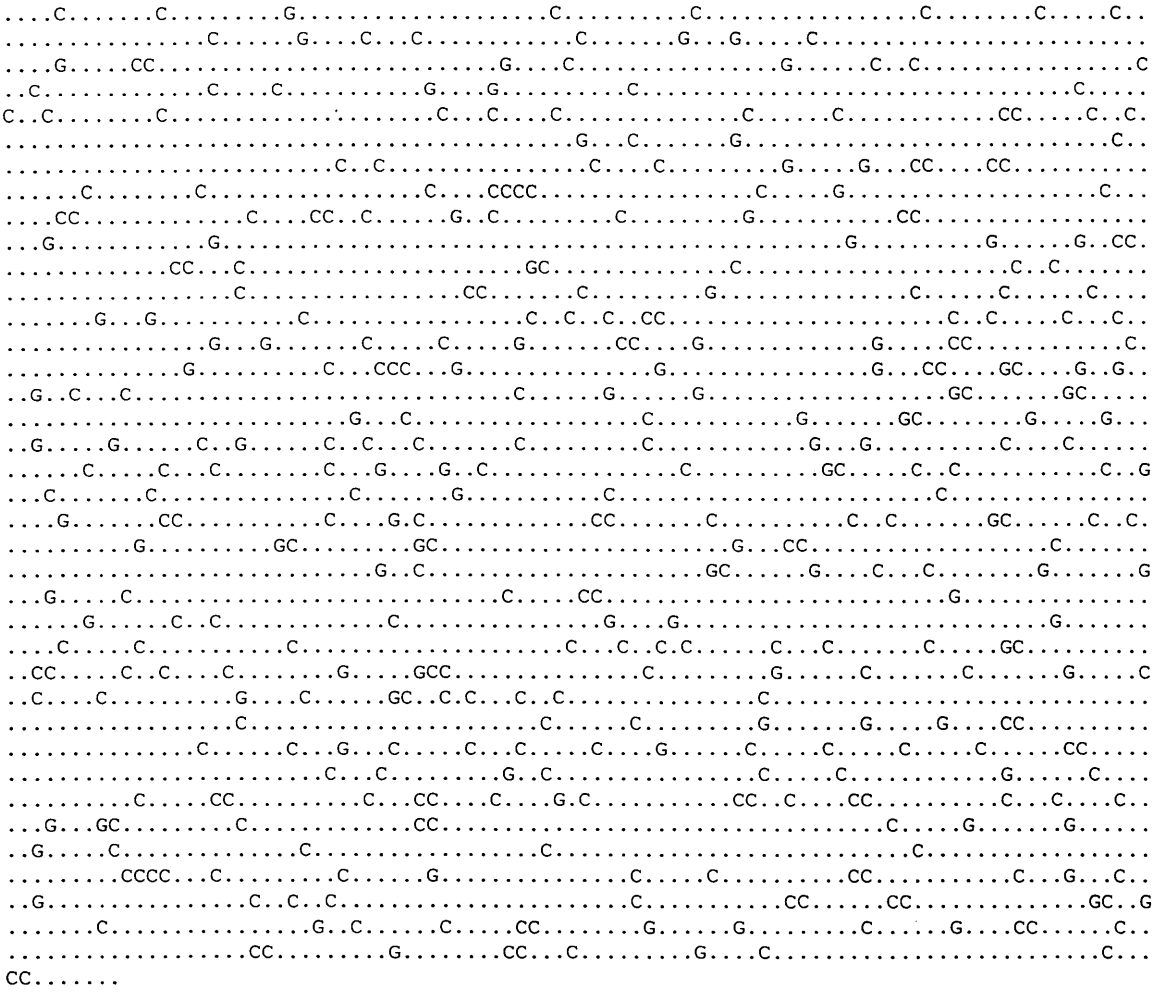


Figure 8.28: Bases counted for feature sem_tt_0.48 for sequence representative of Class 2. These represent 11.3% of the sequence.

Figure 8.29: Bases counted for feature s_ATT_0.12 for sequence representative of Class 1. These represent 53.1% of the sequence.

Figure 8.30: Bases counted for feature s_ATT_0.12 for sequence representative of Class 2. These represent 48.9% of the sequence.

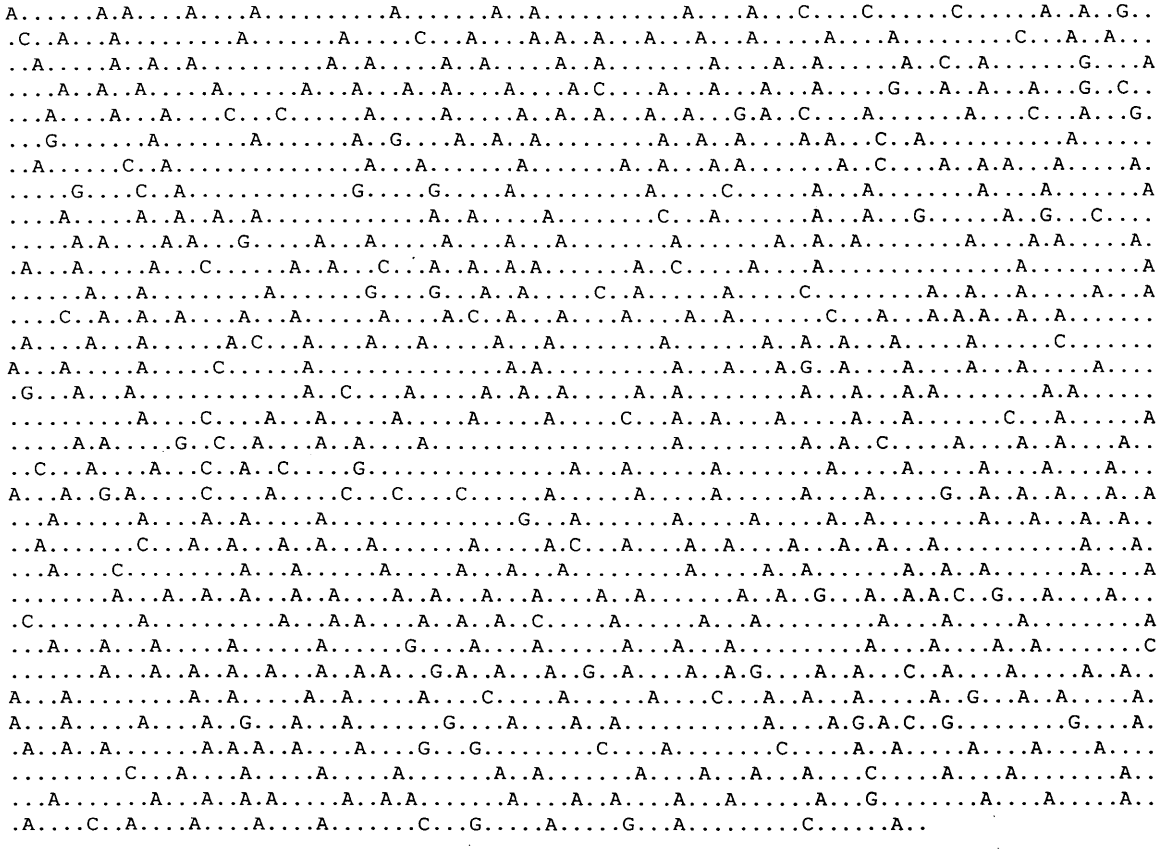


Figure 8.31: Bases counted for feature sem_tt_0.53 for sequence representative of Class 1. These represent 19.4% of the sequence.

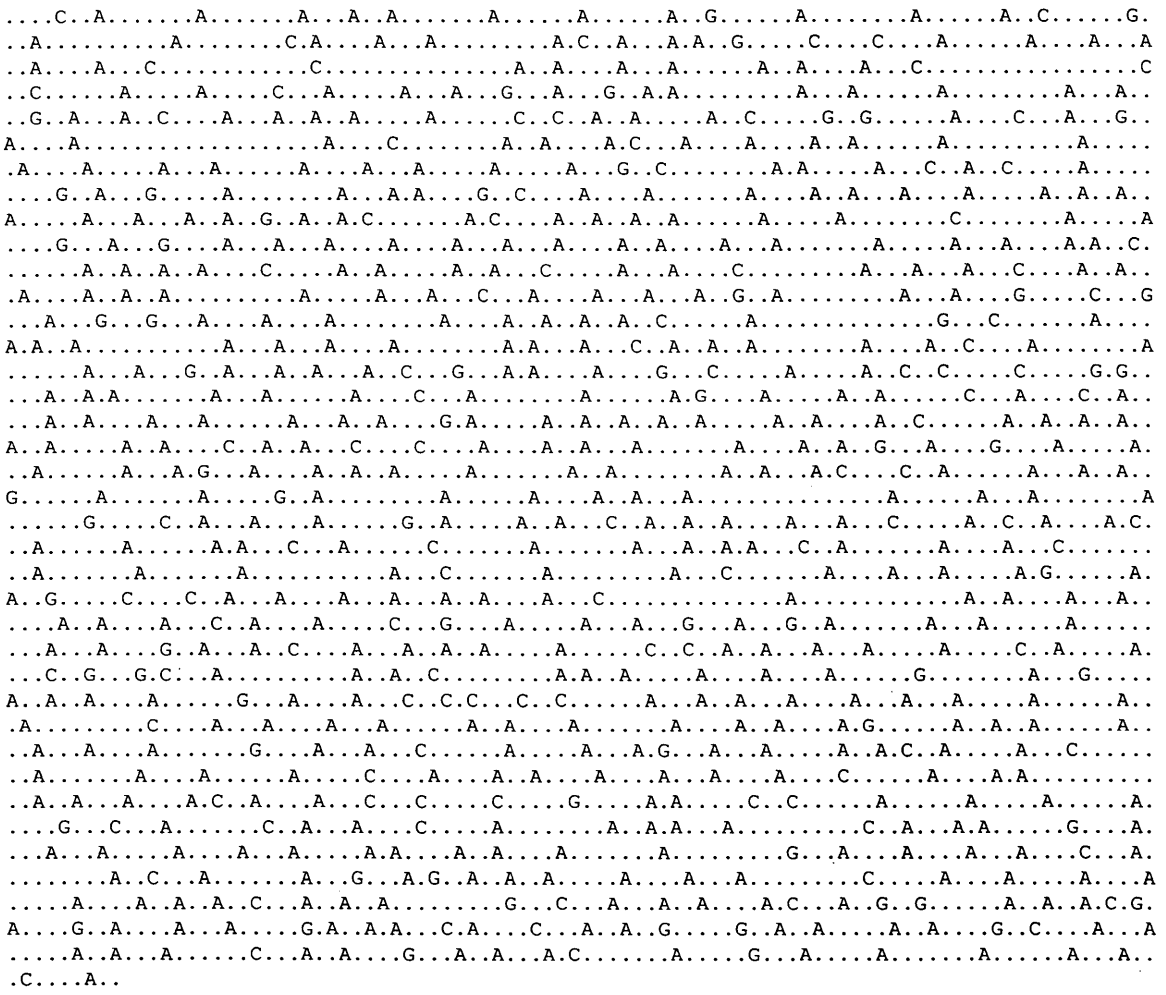


Figure 8.32: Bases counted for feature sem_tt_0.53 for sequence representative of Class 2. These represent 20.5% of the sequence.

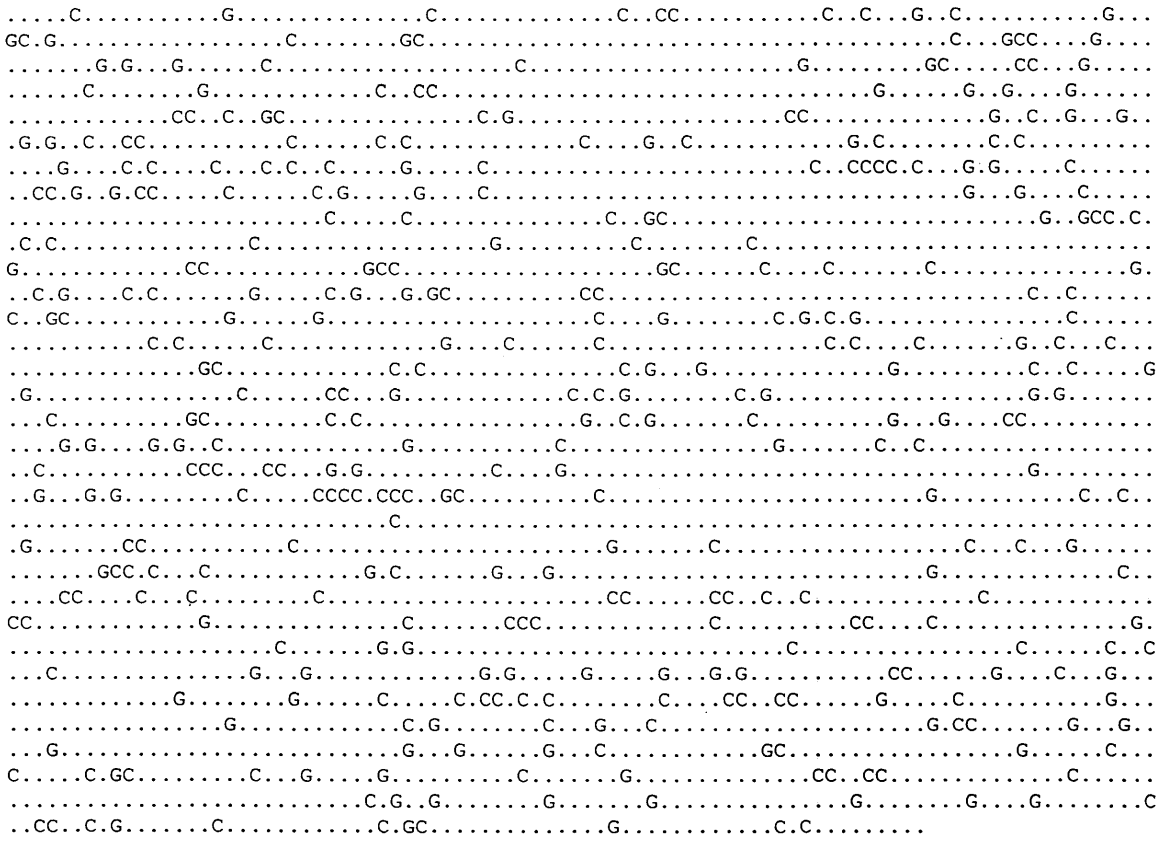


Figure 8.33: Bases counted for feature sem_C_0.06 for sequence representative of Class 1. These represent 12.8% of the sequence.

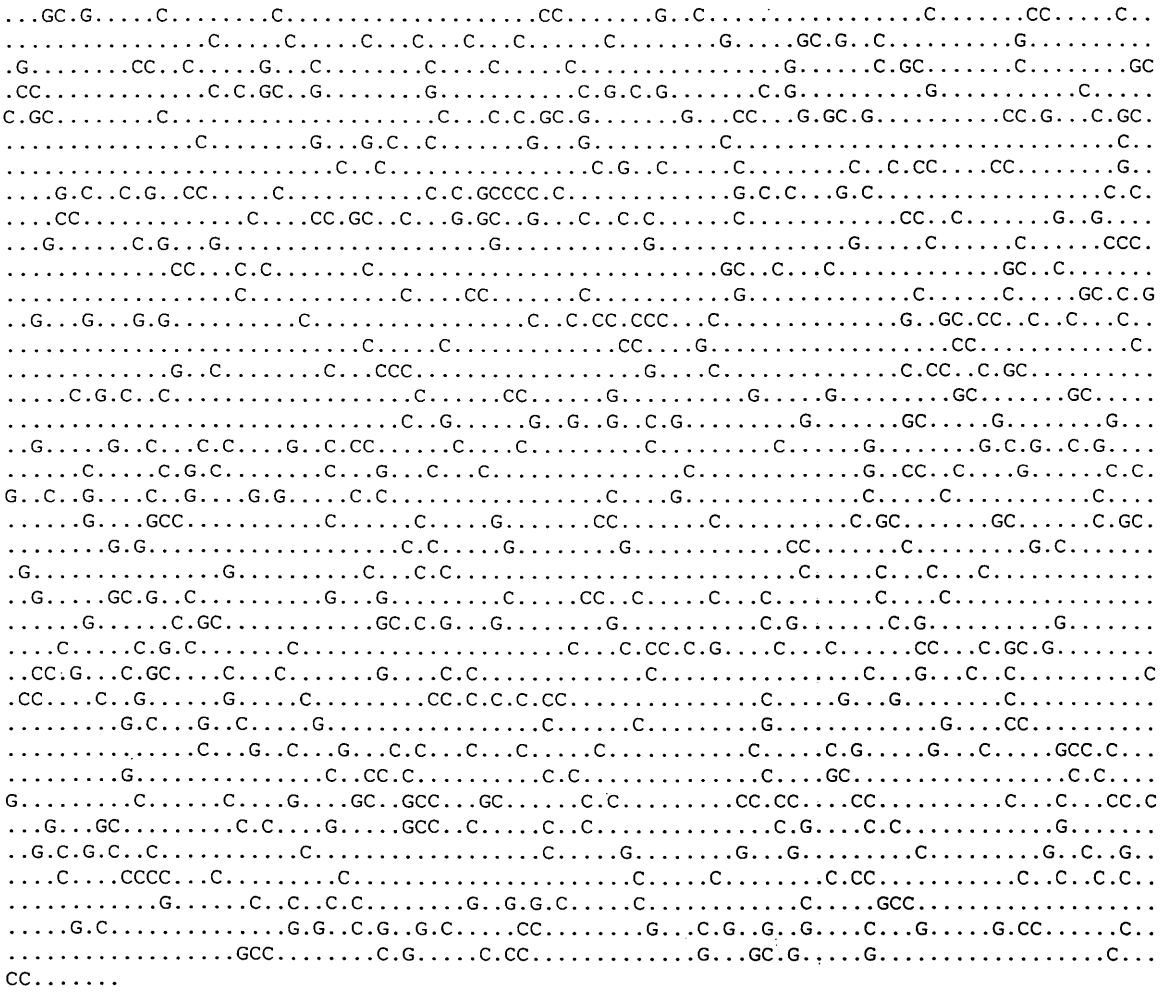


Figure 8.34: Bases counted for feature sem_C_0.06 for sequence representative of Class 2. These represent 17.0% of the sequence.

9 Conclusion

The greatest challenge in applying machine learning techniques to bioinformatics is the reliability of the data. Genome annotations are in a state of flux. Both the sequence and the annotations are constantly being refined and adjusted. This means that training data is always, in some sense, noisy. It also means that there is no dependable way of testing results. The original goal for this research was to build a software tool that would scan for ERVs with adjustable parameters to accommodate peculiarities of different organisms. That goal had to be scaled down, however, due to the unreliability of the data. What was possible to achieve is described in Chapter 7. The excellent annotations for the *Drosophila* genome allowed for some progress.

A substantial impediment to progress was the lack of standards for genome annotation. Biologists tend to specialize on individual organisms: different terminology, different standards, and different methods are used for different types of organisms. As genome assemblies are constantly being updated, genome annotations must be as well. The current system requires a great deal of manual effort for this. This results in situations in which previously assembled databases go off line, as HERVd and RetroSearch

did in the course of this work. There needs to be both more automation and more standardization. This is critical given the massive explosion of genomic data and requires contributions by both biologists and computer scientists. This thesis is such a contribution.

The majority of this thesis is focused on designing sequence features that can be used to refine biological understanding of genomic elements. The statistical features described in Chapter 3 demonstrate how knowledge of computational techniques can be paired with biological knowledge to create new useful tools. In particular the features described in Sections 3.1 and 3.2 combine knowledge of the usefulness of the Fourier transform in detecting changes in patterns with the biological knowledge that ERVs make an unusual use of reading frames.

In Chapter 4 the new technology of side effect machine features is studied, their fitness landscape analyzed, and innovations to their use are introduced. These include new fitness functions and a new method for incorporating SEM features into effective classifiers. This new method involves a novel type of feature selection, dissimilarity clustering, that could be applied to other problems that require feature selection from a set of features, already selected for quality, that are correlated with each other. Another SEM innovation, introduced in Section 6.4, explores the use of SEMs with transitions other than the previously used A-C-G-T transitions.

Chapter 6 uses the features from Chapters 3 and 4 to build classifiers for various

classification problems involving TEs. These are effective for distinguishing retroviruses from genes and from non-coding sequences, for distinguishing different types of viruses from each other, for distinguishing different types of TEs from each other (solitary LTRs and SINEs), and for distinguishing IESs from MDSs in *Tetrahymena* sequences.

Side effect machines provide a computational tool that can be used to create an environment conducive to developing hypotheses about sequences to be later tested experimentally. Chapters 5 and 8 explain how this could work with SEM features. Future work will expand the ideas in these chapters through collaboration with biologists.

The greatest flaw in this thesis is its lack of coherence and unity. All the research in it is related to the problem of genome annotation, but the connections between the pieces is, at times, tenuous. This was due to a lack of understanding when the project was begun of the difficulties to be encountered. The research was started with little biological knowledge of the issues involved and with an assumption that data quality would be better than it actually was. As the research proceeded, the need to fill in details became clear. For example, when false positives were encountered in the scan of the human genome resulting from confusion between solitary LTRs and SINEs, it was necessary to find some way to distinguish the two. Surprisingly, existing methods were inadequate and the problem was both challenging and interesting.

An important contribution computational scientists can make towards solving the difficult problems encountered by biologists is to provide improved description. The

features introduced in this thesis do just that. It is also important to be able to take that description and translate it into something biologically meaningful. The development of techniques for using and interpreting side effect machines increases their value as descriptive tools. The work of annotating and understanding the function of genomes is the type of scientific problem that fits Alfred Lord Tennyson's description in his poem *Ulysses*: "that untraveled world whose margin fades forever and forever as we move." The more we learn, the more we discover there is to learn.

Bibliography

[1] L. Aagaard, P. Villesen, A.L. Kjelbjerg, and F.S. Pedersen. The $\tilde{3}$0-million-year-old ervpb1 envelope gene is evolutionarily conserved among hominoids and old world monkeys. *Genomics*, 86:685–691, 2005.

[2] G. Abrusan, N. Grundmann, L. DeMester, and W. Makalowski. TEclass – a tool for automated classification of unknown eukaryotic transposable elements. *Bioinformatics*, 25(10):1329–1330, 2009.

[3] K. Ahn and H. Kim. Structural and quantitative expression analyses of herv gene family in human tissue. *Molecules and Cells*, 28:99–103, 2009.

[4] M. Akhtar, J. Epps, and E. Ambikairajah. Signal processing in sequence analysis: Advances in eukaryotic gene prediction. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):310–321, 2008.

[5] C. Allauzen, C. Cortes, and M. Mohri. Large-scale training of SVMs with automata kernels. In *International Conference on Implementation and Application of Automata*, pages 17–27, Berlin, 2011. Springer.

[6] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[7] D. Anastassiou. Genomic signal processing. *IEEE Signal Processing Magazine*, 18(4):8–20, 2001.

[8] A.E. Armitage, A. Katzourakis, T. de Oliveira, J.J. Welch, R. Belshaw, K.N. Bishop, B. Kramer, A.J. McMichael, A. Rambaut, and A.K. Iversen. Conserved footprints of APOBEC3G on hypermutated human immunodeficiency virus type 1 and human endogenous retrovirus HERV-K(HML2) sequences. *Journal of Virology*, 82(17):8743–8761, 2008.

[9] D. Ashlock, C. Kussela, and N. Rogers. Hormonal systems for prisoners dilemma agents. In *IEEE Conference on Computational Intelligence and Games*, pages 63–70, 2011.

[10] D. Ashlock and A. McEachern. Ring optimization of side effect machines. In *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 19, pages 165–172, 2009.

[11] D. Ashlock and A. McEachern. Nearest neighbor training of side effect machines for sequence classification. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8, 2010.

[12] D. Ashlock and N. Rogers. The impact of long term memory in the iterated prisoner's dilemma. In *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 19, pages 245–252, 2009.

[13] D. Ashlock and E. Warner. Classifying synthetic and biological DNA sequences with side effect machines. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 22–29, 2008.

[14] W. Ashlock and S. Datta. Detecting retroviruses using reading frame information and side effect machines. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8, 2010.

[15] W. Ashlock and S. Datta. Fast algorithms for recognizing retroviruses. In *Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics*, pages 1–4, 2010.

[16] W. Ashlock and S. Datta. Using Fourier phase analysis on genomic sequences to identify retroviruses. In *Proceedings of the ACM International Conference on Bioinformatics and Computational Biology*, pages 406–409, 2010.

[17] W. Ashlock and S. Datta. Distinguishing endogenous retroviral long terminal repeats from SINE elements using side effect machines. *IEEE/ACM Transactions on Bioinformatics and Computational Biology*, 9:1676–1689, 2012.

[18] W. Ashlock and S. Datta. Evolved features for DNA sequence classification and their fitness landscapes. *IEEE Transactions on Evolutionary Computation*, 17:185–197, 2013.

[19] R. Belshaw, V. Pereira, A. Katzourakis, G. Talbot, Jan Paces, A. Burt, and M. Tristem. Long-term reinfection of the human genome by endogenous retroviruses. *Proceedings of the National Academy of Sciences of the USA*, 101(14):4894–4899, 2004.

[20] F. Benachenhou, P. Jern, M. Oja, G. Sperber, V. Bilkstad, P. Somervuo, S. Kaski, and J. Blomberg. Evolutionary conservation of orthoretroviral long terminal repeats (LTRs) and *ab initio* detection of single LTRs in genomic data. *PLoS ONE*, 4(4):e5179, 2009.

[21] C.M. Bergman and H. Quesneville. Discovering and detecting transposable elements in genome sequences. *Briefings in Bioinformatics*, 8(6):382–392, 2007.

[22] Pedro Bernaola-Galvan, Ivo Grosse, Pedro Carpena, Jose L. Oliver, Ramon Roman-Roldan, and H. Eugene Stanley. Finding borders between coding and noncoding DNA regions by an entropic segmentation method. *Physical Review Letters*, 85:1342–1345, 2000.

[23] V. Blikstad, F. Benachenhou, G.O. Sperber, and J. Blomberg. Evolution of human endogenous retroviral sequences: a conceptual account. *Cellular and Molecular Life Sciences*, 65:3348–3365, 2008.

[24] G.W. Blissard and G.F. Rohrmann. Baculovirus diversity and molecular biology. *Annual Review of Entomology*, 35:127–155, 1990.

[25] S. Bochkanov and V. Bystritsky. *alglib*, 1999-2010.

[26] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[27] Broad Institute of Harvard and MIT. Tetrahymena comparative sequencing project. `http://www.broadinstitute.org`.

[28] J. Brown, S. Houghten, and D. Ashlock. Side effect machines for quaternary edit metric decoding. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8, 2010.

[29] S. Burge, G.N. Parkinson, P. Hazel, A.K. Todd, and S. Neidle. Quadruplex DNA: sequence, topology and structure. *Nucleic Acids Research*, 34(19):5402–5415, 2006.

[30] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[31] A. Buzdin, E. Kovalskaya-Alexandrova, E. Gogvadze, and E. Sverdlow. At least 50 percent of human-specific HERV-K (HML-2) long terminal repeats serve in vivo as active promoters for host nonrepetitive DNA transcription. *Journal of Virology*, 80(21):10752–10762, 2006.

[32] A. Capsi and L. Pachter. Identification of transposable elements using multiple alignments of related genomes. *Genome Research*, 16:260–270, 2006.

[33] N. Chakravarthy, A. Spanias, L.D. Iasemidis, and K. Tsakalis. Autoregressive modeling and feature analysis of DNA sequences. *EURASIP Journal on Applied Signal Processing*, 1:13–28, 2004.

[34] D.L. Chalker and M. Yao. DNA elimination in ciliates: transposon domestication and genome surveillance. *Annual Review of Genetics*, 45:227–46, 2011.

[35] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[36] C. Cheng, A. Vogt, K. Mochizuki, and M. Yao. A domesticated *piggybac* transposase plays key roles in heterochromatin dynamics and DNA cleavage during programmed DNA deletion in *Tetrahymena thermophila*. *Molecular Biology of the Cell*, 21:1753–1762, 2010.

[37] Y.C. Chew, J.T. West, S.J. Kratzer, A.M. Ilvarsonn, J.C. Eissenberg, B.J. Dave, D. Klinkebiel, J.K. Christman, and J. Zempleni. Biotinylation of histones represses transposable elements in human and mouse cells and cell lines and in *Drosophila melanogaster*. *The Journal of Nutrition*, 138(12):2316–2322, 2008.

[38] Robert S. Coyne. personal communication.

[39] R.S. Coyne, M. Lhuillier-Akakpo, and S. Duharcourt. RNA-guided DNA rearrangements in ciliates: Is the best genome defence a good offence? *Biology of the Cell*, 104:309–325, 2012.

[40] Suprakash Datta and Amir Asif. A fast DNA based gene prediction algorithm for identification of protein coding regions. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 653–656, 2005.

[41] M. Dewannieux, F. Harper, A. Richaud, C. Letzelter, D. Ribet, G. Pierron, and T. Heidmann. Identification of an infectious progenitor for the multiple-copy herv-k human endogenous retroelements. *Genome Research*, 16(12):1548–56, 2006.

[42] D.A. Dramerov and N.S. Vassetzky. SINEs. *Wiley Interdisciplinary Reviews: RNA*, pages 772–786, 2011.

[43] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1999.

[44] S.R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.

[45] J.A. Eisen, R.S. Coyne, M. Wu, D. Wu, M. Thiagarajan, et al. Macronuclear genome sequence of the ciliate *Tetrahymena thermophila*, a model eukaryote. *PLoS Biology*, 4(9):e286, 2006.

[46] N.V. Federoff. Transposable genetic elements in maize. *Scientific American*, 250(6):84–98, 1984.

[47] C. Feschotte, U. Keswani, N. Ranganathan, M. Guibotsy, and D. Levine. Exploring repetitive DNA landscapes using REPCLASS, a tool that automates the classification of transposable elements in eukaryotic genomes. *Genome Biology and Evolution*, 1:205–220, 2009.

[48] J.S. Fillingham, T.A. Thing, N. Vythilingum, A. Keuroghlian, D. Bruno, G.B Golding, and R.E. Pearlman. A non-long terminal repeat retrotransposon family is restricted to the germ line micronucleus of the ciliated protozoan Tetrahymena thermophila. *Eukaryotic Cell*, 1:157–169, 2004.

[49] M.J. Fraser. The TTAA-specific family of transposable elements. In A.A. James and A.H. Handler, editors, *Insect Transgenesis: Methods and Applications*. CRC Press, 2000.

[50] R. Gifford and M. Tristem. The evolution, distribution and diversity of endogenous retroviruses. *Virus Genes*, 26(3):291–315, 2003.

[51] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982.

[52] KD Pruittand J Harrow, RA Harte, C Wallin, M Diekhans, DR Maglott, S Searle, CM Farrell, JE Loveland, BJ Ruef, E Hart, MM Suner, MJ Landrum, B Aken, S Ayling, R Baertsch, J Fernandez-Banet, JL Cherry, V Curwen, M Dicuccio, M Kellis, J Lee, MF Lin, M Schuster, A Shkeda, C Amid, G Brown, O Dukhanina, A Frankish, J Hart, BL Maidak, J Mudge, MR Murphy, T Murphy, J Rajan, B Rajput, LD Riddick, C Snow, C Steward, D Webb, JA Weber, L Wilming, W Wu, E Birney, D Haussler, T Hubbard, J Ostell, R Durbin, and D Lipman. The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes. *Genome Research*, 19(7):1316–23, 2009.

[53] RA Harte, CM Farrell, JE Loveland, MM Suner, L Wilming, B Aken, D Barrell, A Frankish, C Wallin, S Searle, M Diekhans, J Harrow, and KD Pruitt. Tracking and coordinating an international curation effort for the CCDS project. *Database*, 20:bas008, 2012.

[54] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning*. Springer, New York, 2009.

[55] M.H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley and Sons, Inc., 1996.

[56] J. Henderson, S. Salzberg, and K.H. Fasman. Finding genes in DNA with a hidden markov model. *Journal of Computational Biology*, 4(2):127–141, 1997.

[57] A. Huda, N. Polavarapu, I.K. Jordan, and J. F. McDonald. Endogenous retroviruses of the chicken genome. *Biology Direct*, 3(9), 2008.

[58] J.F. Hughes and J.M. Coffin. Human endogenous retrovirus K solo-LTR formation and insertional polymorphisms: Implications for human and viral evolution. *PNAS*, 101(6):1668–1672, 2004.

[59] P. Huvos. A member of a repeat family is the source of an insertion-deletion polymorphism inside a developmentally eliminated sequence of *Tetrahymena thermophila*. *Journal of Molecular Biology*, 336:1061–1073, 2004.

[60] P. Jern and J.M. Coffin. Effects of retroviruses on host genome function. *Annual Review of Genetics*, 42:709–32, 2008.

[61] P. Jern, G. Sperber, and J. Blomberg. Use of endogenous retroviral sequences (ervs) and structural markers for retroviral phylogenetic inference and taxonomy. *Retrovirology*, 2(50), 2005.

[62] P. Jern, J.P. Stove, and J.M. Coffin. Role of APOBEC3 in genetic diversity among endogenous murine leukemia viruses. *PLoS Genetics*, 3(10):e183, 2007.

[63] N.C. Jones and P.A. Pavzner. *An Introduction to Bioinformatics Algorithms*, chapter Hidden Markov Models. The MIT Press, 2004.

[64] N. Juretic, T.E. Bureau, and R.M. Bruskiewich. Transposable element annotation of the rice genome. *Bioinformatics*, 20(2):155–160, 2004.

[65] J. Jurka, V.V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase update, a database of eukaryotic repetitive elements. *Cytogentic and Genome Research*, 110:462–467, 2005.

[66] A. Kalyanaraman and S. Aluru. Efficient algorithms and software for detection of full-length LTR retrotransposons. *Journal of Bioinformatics and Computational Biology*, 4(2):197–216, 2006.

[67] U. Kamath, A. Shehu, and K. De Jong. Using evolutionary computation to improve SVM classification. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[68] U. Kamath, A. Shehu, and K.A. De Jong. A two-stage evolutionary approach for effective classification of hypersensitive DNA sequences. *Journal of Bioinformatics and Computational Biology*, 9(3):399–413, 2011.

[69] M. Kargar and A. An. Evaluation of different complexity measures for signal detection in genome sequences. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 422–425, 2010.

[70] A.L. Kjelbjerg, P. Villesen, L. Aagaard, and F.S. Pedersen. Gene conversion and purifying selection of a placenta-specific erv-v envelope gene during simian evolution. *BMC Evolutionary Biology*, 8(266), 2005.

[71] LA Klobutcher and G. Herrick. Developmental genome reorganization in ciliated protozoa: the transposon link. volume 56 of *Progress in Nucleic Research and Molecular Biology*, pages 1–62. Academic Press, 1997.

[72] D. Kotlar and Y. Lavner. Gene prediction by spectral rotation measure: A new method for identifying protein-coding regions. *Genome Research*, 13:1930–1937, 2003.

[73] A. Krogh. Gene finding: putting the parts together. In M. Bishop, editor, *Guide to Human Genome Computing*, pages 261–274. Academic Press, 1998.

[74] S. Kurtz, J.V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29(22):4633–4642, 2001.

[75] S. Kurtz and C. Schleiermacher. Reputer: fast computation of maximal repeats in complete genomes. *Bioinformatics*, 15(5):426–427, 1999.

[76] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, 7:566–575, 2002.

[77] B. Lewin. *Genes VII*. Oxford University Press, New York, 2000.

[78] Benjamin Lewin. *Genes VII*. Oxford University Press, New York, 2000.

[79] R. Li, J. Ye, S. Li, J. Wang, Y. Han, C. Ye, J. Wang, H. Yang, J. Yu, G.K. Wong, and J. Wang. ReAS: Recovery of ancestral sequences for transposable elements from the unassembled reads of a whole genome shotgun. *PLoS Computational Biology*, 1:e43, 2005.

[80] J. Ling, W. Pi, D. Tuan, et al. The solitary long terminal repeats of ERV-9 endogenous retrovirus are conserved during primate evolution and possess enhancer activities in embryonic and hematopoietic cells. *Journal of Virology*, 76(5):2410–2423, 2002.

[81] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–41, 1985.

[82] J.V. Lorenzo-Ginori, A. Rodriguez-Fuentes, R.G. Abalo, and R.S. Rodriguez. Digital signal processing in the analysis of genomic sequences. *Current Bioinformatics*, 4:28–40, 2009.

[83] C.B. Lowe and D. Haussler. 29 mammalian genomes reveal novel exaptations of mobile elements for likely regulatory functions in the human genome. *PLoS one*, 7:e43128, 2012.

[84] R. Lower, J. Lower, and R. Kurth. The viruses in all of us: Characteristics and biological significance of human endogenous retrovirus sequences. *Proceedings of the National Academy of Sciences USA*, 93:5117–5184, 1996.

[85] J. Ma, K.M. Devos, and J.L. Bennetzen. Analyses of LTR-retrotransposon structures reveal recent and rapid genomic DNA loss in rice. *Genome Research*, 14:860–869, 2004.

[86] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[87] M. Maechler, P. Rousseeuw, Struyf A, Hubert M, and Hornik K. *cluster: Cluster Analysis Basics and Extensions*, 2011. R package version 1.14.1.

[88] S. Mahfoud. Niching methods for genetic algorithms. Technical report, 1995.

[89] V. Makarov. Computer programs for eukaryotic gene prediction. *Briefings in Bioinformatics*, 3(2):195–199, 2002.

[90] H. Masoom, S. Datta, A. Asif, L. Cunningham, and G. Wu. A fast algorithm for detecting frame shifts in DNA sequences. In *CIBCB*, pages 1–8, 2006.

[91] E. M. McCarthy and J. F. McDonald. LTR_STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics*, 19(3):362–367, 2003.

[92] F. Murtagh. *Multidimensional Clustering Algorithms*. Wuerzburg: Physica-Verlag, 1985.

[93] E.W. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, M.J. Flanigan, S.A. Kravitz, C.M. Mobarry, K.H.J. Reinert, K.A. Remington, E.L. Anson, R.A. Bolanos, H. Chou, C.M. Jordan, A.L. Halpern, S. Lonardi, E.M. Beasley, R.C. Brandon, L. Chen, P.J. Dunn, Z. Lai, Y. Liang, D.R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G.M. Rubin, M.D. Adams, and J.C. Venter. A whole-genome assembly of Drosophila. *Science*, 287:2196–2204, 2000.

[94] Z. Ning, A.J. Cox, and J.C. Mullikin. Ssaha: A fast search method for large DNA databases. *Genome Research*, 11:1725–1729, 2001.

[95] H. Nishihara, AFA Smit, and N. Okada. Functional noncoding sequences derived from SINEs in the mammalian genome. *Genome Research*, 16(7):864–874, 2006.

[96] F.A. Noori and S. Houghten. A multi-objective genetic algorithm with side effect machines for motif discovery. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 275–282, 2012.

[97] K.R. Oliver and W.K. Greene. Transposable elements: powerful facilitators of evolution. *Bioessays*, 31:703–714, 2009.

[98] J. Paces, A. Pavlicek, and V. Paces. HERVd: database of human endogenous retroviruses. *Nucleic Acids Research*, 30(1):205–6, 2002.

[99] J. Paces, A. Pavlicek, R. Zika, V.V. Kapitonov, J. Jurka, and V. Paces. HERVd: the human endogenous retroviruses database: update. *Nucleic Acids Research*, 32:D50, 2004.

[100] A. Pavlicek, J. Paces, D. Elleder, and J. Hejnar. Processed pseudogenes of human endogenous retroviruses generated by lines: Their integration, stability, and distribution. *Genome Research*, 12:391–399, 2002.

[101] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the USA*, 85(8):2444–8, 1988.

[102] P. Perot, N. Mugnier, C. Montgiraud, J. Gimenez, M. Jaillard, B. Bonnaud, and F. Mallet. Microarray-based sketches of the HERV transcriptome landscape. *PLoS ONE*, 7(6):e40194, 2012.

[103] W. Pi et al. Long-range function of an intergenic retrotransposon. *PNAS*, 107:12992–12997, 2010.

[104] B. Pierce. *Genetics: A conceptual approach*. W.H. Freeman and Company, New York, 2005.

[105] M. Pop, S.L. Salzberg, and M. Shumway. Genome sequence assembly: Algorithms and issues. *IEEE Computer*, pages 47–54, 2002.

[106] A.L. Price, N.C. Jones, and P.A. Pevzner. *De novo* indentification of repeat families in large genomes. *Bioinformatics*, 21:i351–i358, 2005. Suppl. 1.

[107] H. Quesneville et al. Combined evidence annotation of transposable elements in genome sequences. *PLoS Computational Biology*, 1(2):e22, 2005.

[108] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[109] K.B. Ramesh, P. Shankar, B.P. Mallikarjunaswamy, and E.T. Puttaiah. Genomic signal processing (GSP) of rheumatic arthritis (RA) using different indicator sequences. *International Journal of Computer Science and Mobile Computing*, 2(5):332–337, 2013.

[110] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[111] M. Rho et al. *De novo* identification of LTR retrotransposons in eukaryotic genomes. *BMC Genomics*, 8(90), 2007.

[112] M.T. Romanish, C.J. Cohen, and D.L. Mager. Potential mechanisms of endogenous retroviral-mediated genomic instability in human cancer. *Seminars in Cancer Biology*, 20:246–253, 2010.

[113] A. Rushdi and J. Tuqan. The role of the symbolic-to-numerical mapping in the detection of DNA periodicities. In *Proc. IEEE Int. Workshop Genomic Signal Processing Statistics*, pages 1–4, 2008.

[114] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[115] S. Saha et al. Empirical comparison of ab initio repeat finding programs. *Nucleic Acids Research*, 36(7):2284–2294, 2008.

[116] K. Scheibye-Alsing et al. Sequence assembly. *Computational Biology and Chemistry*, 33:121–136, 2009.

[117] J. Schonfeld and D. Ashlock. Classifying Cytochrome c Oxidase subunit 1 by translation initiation mechanism using side effect machines. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–7, 2010.

[118] D. Shakya, R. Saxena, and S. Sharma. An adaptive window length strategy for eukaryotic CDS prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, PP(99), 2013.

[119] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423; 623–656, 1948.

[120] M. Sipser. *Introduction to the Theory of Computation*. Thomson, second edition, 2006.

[121] AFA Smit, R. Hubley, and P. Green. RepeatMasker Open-3.0, 1996-2004. <http://www.repeatmasker.org>.

[122] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[123] G. Sperber, A. Lovgren, N. Eriksson, F. Benachenhou, and J.Blomberg. Retrotector online, a rational tool for analysis of retroviral elements in small and medium size vertebrate genomic sequences. *BMC Bioinformatics*, 10(S4), 2009.

[124] G. O. Sperber, T. Airola, P. Jern, and J. Blomberg. Automated recognition of retroviral sequences in genomic data – RetroTector. *Nucleic Acids Research*, 35:4964–4976, 2007.

[125] S. Steinbiss, U. Willhoeft, G. Gremme, and S. Kurtz. Fine-grained annotation and classification of *de novo* predicted LTR retrotransposons. *Nucleic Acids Research*, 37(21):7002–7013, 2009.

[126] J.P. Stoye. Endogenous retroviruses: still active after all these years? *Current Biology*, 11:R914–R916, 2001.

[127] F. Sun et al. Common evolutionary trends for SINE RNA structures. *TRENDS in Genetics*, 23(1), 2006.

[128] E.D. Sverdlov. Perpetually mobile footprints of ancient infections in human genome. *FEBS Letters*, 428:1–6, 1998.

[129] V. Svetnik, A. Liaw, C. Tong, and T. Wang. Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In *Multiple Classifier Systems*, pages 334–343, Berlin, 2004. Springer.

[130] Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning*, 2012. R package version 4.1-0.

[131] S. Tiwari, S. Ramachandran, A. Bhattacharya, and R. Ramaswamy. Prediction of probable genes by Fourier analysis of genomic sequences. *Computer Applications in the Biosciences*, 13(3):263–270, 1997.

[132] E.N. Trifonov. 3-, 10.5-, 200- and 400-base periodicities in genome sequences. *Physica A*, 249:511–516, 1998.

[133] Jamal Tuqan and Ahmad Rushdi. A DSP approach for finding the codon bias in DNA sequences. *IEEE Journal of Selected Topics in Signal Processing*, 2:343–356, 2008.

[134] S. Tweedle, M. Ashburner, K. Fall, P. Leyland, P. McQuilton, S. Marygold, G. Millburn, D. Osumi-Sutherland, A. Schroeder, R. Seal, H. Zhang, and The FlyBase Consortium. FlyBase: enhancing Drosophila gene ontology annotations. *Nucleic Acids Research*, 37:D555–D559.

[135] Bradley University. Tetrahymena Genome Database, 2004-. <http://www.ciliate.org>.

[136] Howard B. Urnovitz and William H. Murphy. Human endogenous retroviruses: nature, occurrence, and clinical implications in human disease. *Clinical Microbiology Reviews*, 9:72–99, 1996.

[137] P. P. Vaidyanathan. Genomics and proteomics: A signal processor's tour. *IEEE Circuits and Systems Magazine*, 4:6–29, 2004.

[138] Luis P. Villarreal. *Viruses and the Evolution of Life*. ASM Press, Washington, D.C., 2005.

[139] P. Villesen, L. Aagaard, C. Wiuf, and F. S. Pedersen. Identification of endogenous retroviral reading frames in the human genome. *Retrovirology*, 1(32):1–13, 2004.

[140] C. Vitte and O. Panaud. Formation of solo-LTRs through unequal homologous recombination counterbalances amplifications of LTR retrotransposons in rice *oryza sativa* l. *Molecular Biology and Evolution*, 20(4):528–540, 2003.

[141] R.F. Voss. Evolution of long-range fractal correlations and $\frac{1}{f}$ noise in DNA base sequences. *Physical Review Letters*, 68(25):3805–3808, 1992.

[142] S. Wain-Hobson. Retrovirus evolution. In E. Domingo et al., editors, *Origin and Evolution of Viruses*, pages 259–278. Elsevier Ltd., New York, 2008.

[143] L. Wang and D. Schonfeld. Mapping equivalence for symbolic sequences: theory and applications. *IEEE Transactions on Signal Processing*, 57(12):4895–4905, 2009.

[144] R.A. Weiss. The discovery of endogenous retroviruses. *Retrovirology*, 3:67, 2006.

[145] R.A. Weiss and P.K. Vogt. 100 years of Rous sarcoma virus. *The Journal of Experimental Medicine*, 208(12):2351–5, 2011.

[146] D.M. Witten and R. Tibshirani. *sparcl: Perform sparse hierarchical clustering and sparse k-means clustering*, 2010. R package version 1.0.1.

[147] J.D. Wultschick, J.A. Gershan, A.J. Lochowicz, S. Li, and K.M. Karrer. A novel family of mobile genetic elements is limited to the germline genome in *Tetrahymena thermophila*. *Nucleic Acids Research*, 30:2524–2537, 2002.

[148] Z. Xu and H. Wang. LTR_FINDER: an efficient tool for the prediction of full-length LTR retrotransposons. *Nucleic Acids Research*, 35:W265–W268, 2007.

[149] M. Yandell and D. Ence. A beginners guide to eukaryotic genome annotation. *Nature Reviews: Genetics*, 13:329–342, 2012.

[150] O. Yavuz and L. Ozyilmaz. Analysis and classification of HIV-1 sub-type viruses by AR model through artificial neural networks. *World Academy of Science, Engineering and Technology*, 49:826–830, 2009.

[151] B. Yoon. Hidden markov models and their applications in biological sequence analysis. *Current Genomics*, 10(6):402–415, 2009.

[152] Chun-Ting Zhang, Ren Zhang, and Hong-Yu Ou. The Z curve database: a graphic representation of genome sequences. *Bioinformatics*, 19(5):593–599, 2003.

[153] Zhi-Ming Zheng and Carl C. Baker. Papillomavirus genome structure, expression, and post-transcriptional regulation. *Front. BioSci.*, 11:2286–2302, 2006.

[154] H. Zhou, L. Du, and H. Yan. Detection of tandem repeats in DNA sequences based on parametric spectral estimation. *IEEE Transactions on Information Technology in Biomedicine*, 13(5):747–755, 2009.