Predicting RNA Binding Proteins Using Discriminative Methods

Shweta Bhandare

University of Colorado at Boulder, bhandare@colorado.edu
Predicting RNA binding proteins using discriminative methods

by

Shweta Bhandare

M.S. Department of Interdisciplinary Telecommunications, University of Colorado at Boulder 2003

B.E., College of Engineering, University of Goa, 1995

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2017
This thesis entitled: Predicting RNA binding proteins using discriminative methods written by Shweta Bhandare has been approved for the Department of Computer Science

Prof. Robin Dowell

Dr. Debra S. Goldberg

Prof. Larry E. Hunter

Dr. Daniel Weaver

Date __________________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
This thesis examines the role of computational methods in identifying the motifs utilized by RNA-binding proteins (RBPs). RBPs play an important role in post-transcriptional regulation and identify their targets in a highly specific fashion through recognition of primary sequence and/or secondary structure hence making the prediction a complex problem.

I applied the existing $k$-spectrum kernel method to a support vector machine and verified the published binding sites of two RBPs: Human antigen R (HuR) and Tristetraprolin (TTP). These RBPs exhibit opposing effects to the bound messenger RNA (mRNA) transcript but have similar binding preferences. Additional feature engineering highlighted the U-rich binding preference for HuR and AU-rich binding preference for TTP. I extended the $k$-spectrum kernel method to incorporate domain adaptation and identified binding preferences specific to each RBP as well as identified binding sites that were shared by the RBPs. The predicted $k$-mers correctly identified U/CU-rich $k$-mers specific to HuR and AU-rich $k$-mers specific to TTP, as well as exposed the $k$-mers that were shared by these RBPs.

In order to assess the performance of computational methods used to predict RBP targets, I developed a Python framework called the PySG. This framework generates custom biologically relevant datasets by embedding either primary sequence-based or secondary structure-based motifs followed by benchmarking computational methods on the generated sequences. To test-drive the PySG framework, the $k$-spectrum kernel method and Discriminative Regular Expression Motif Elicitation (DREME) were included in the framework and their performance was benchmarked on generated datasets. Benchmarking results demonstrate that the $k$-spectrum kernel method correctly identifies primary sequence binding sites, whereas DREME performs better when the binding sites involve secondary structure. The $k$-spectrum kernel considers only $k$-mers, which are
indicative of primary sequence, while DREME allows wildcard characters when identifying motifs which implicitly captures the covariation present in secondary structure binding sites. The novelty of the framework is the ability to generate sequences that simulate biological conditions for RBP binding and select an appropriate computational method for the dataset being studied. Thus making the PySG framework a valuable contribution to the field of bioinformatics.
Dedication

To my parents - who taught me by example that there are no secrets to success. It is the result of preparation, hard work, grit, and picking yourself up after every failure.
Acknowledgements

First and foremost, I would like to express my sincere gratitude to my graduate advisor Prof. Robin Dowell who accepted me as her graduate student, laid out a plan that would get me to the finish line, and ensured that I stayed focused on executing to the plan. Her expertise in the field of bio-informatics, systematic approach to problem-solving, ability to see the big picture, and her guidance, when faced with challenging problems, made this thesis a possibility. She stood by me through the various ups and downs, empathized with me, and had a clear plan to tackle the next problem at hand. I always left her office reassured that the plan we had formulated would work and would get me to the goal.

My co-advisor, Dr. Debra S. Goldberg has always been there for me. She was there when I started the process, stood by me through the several difficult periods that I went through while working on this thesis, and finally there when I finished up my thesis. She believed in me and encouraged me at every step of this very long Ph.D. process. I couldn’t thank her enough.

I am very grateful to Prof. Larry Hunter for his encouragement and support through the different phases of this Ph.D. thesis. Despite his busy schedule, he always made time to answer questions I had, provided guidance on solving different problems that I countered, and always offered honest feedback that helped shaped this thesis.

I would like to thank Prof. Jordan Boyd-Graber who provided invaluable machine learning expertise and insights when solving some hard computational biology problems. He has always been prompt in replying to questions either through email or in person. I would like to acknowledge Dr. Daniel Weaver, who readily accepted to be on my thesis committee, meticulously read through the
thesis document, and provided invaluable feedback that helped structure my thesis document.

I am grateful for all the guidance I received from Prof. Asa Ben-Hur from the Colorado State University. He introduced me to the various kernel methods used in the field of bioinformatics, as well as the method that I ended up using for my research. Additionally, he was always there when I needed any guidance towards my thesis.

I would like to take this opportunity to thank Prof. Bradley Olwin, Dr. Nick Farina, and Dr. Crystal Pulliam for working with me during the initial phase of this thesis. They helped me understand the biology concepts relevant to the project. I would also like to acknowledge Dr. Suzanne Gallagher and Dr. Betty Eskow from Dr. Goldberg’s lab who always encouraged me. Most importantly, I would like to thank Dr. David Knox from Dr. Dowell’s lab. He always asked the right questions, helped me review the thesis document, and more importantly was there to brainstorm possible approaches to solving problems. His contribution improved the quality of my thesis.

I would like to thank my co-workers at LogRhythm for being supportive and allowing me the flexibility to work towards my thesis.

I would like to thank all the developers that contribute to www.stackoverflow.com. Their contribution helped speed the development of the Python framework that was an important contribution of this thesis. I would also like to thank the people who contribute to the www.tex.stackexchange.com. Their contribution was invaluable in writing my thesis in Latex.

A huge gratitude goes to my M.S thesis graduate advisor Dr. Timothy X. Brown. He was responsible for instilling a very strong work ethic in me and was there to help me rehearse my Ph.D. thesis defense exam presentation.

Most importantly, I would like to thank my family: my husband, Karthik Sundaresan and kids (Adhrit and Arjun) who were more than supportive while I worked towards my thesis. My in-laws who always showered me with their best wishes, and encouragement. My sister has always been supportive towards my goals and ambitions.
Finally, to my parents, I reserve the highest gratitude. They always encouraged me to dream and then work hard to achieve those dreams. They instilled values of honesty, hard work, determination, and grit by modeling those values in their day to day life.
## Contents

### Chapter

1. **Introduction**
   - 1.1 Thesis Outline .......................................................... 2

2. **Biology Background**
   - 2.1 Brief RNA Primer ....................................................... 5
   - 2.2 RNA binding proteins and their roles in regulation ................. 8
     - 2.2.1 RNA binding domains ................................................. 9
   - 2.3 Experimental methods to identify target RBPs ......................... 12
     - 2.3.1 *In vitro* Methods .................................................. 13
     - 2.3.2 *In vivo* Methods .................................................. 15
   - 2.4 Challenges of analysis of experimental data .......................... 17
   - 2.5 Review of Computational tools for RBP analysis ....................... 18
     - 2.5.1 Motif-Searching Computational Tools ........................... 18
     - 2.5.2 Covariance Models (CMs) ......................................... 23

3. **Computational Background**
   - 3.1 RBP target identification as a classification problem ............. 27
   - 3.2 Support Vector Machines and Kernel Methods ........................ 29
     - 3.2.1 k-spectrum kernel method ..................................... 33
     - 3.2.2 Applications of Kernel Methods in Computational Biology .... 36
5.3.3 Step 3: Creating FASTA files .................................................. 76
5.3.4 Step 4: Running Computational tools ..................................... 79
5.3.5 Step 5: Evaluate Discriminative methods ................................. 80
5.3.6 Step 6: Benchmark results .................................................... 83

5.4 Augmenting the framework ....................................................... 84
5.5 Guidelines for generating sequences using PySG framework ......... 85
5.6 Discussion and Future Work ...................................................... 88
  5.6.1 Validating sequences generated using a model ....................... 89

6 Benchmarking Results ................................................................. 91
  6.1 Selection of signals: primary and secondary structure ................. 91
  6.2 Experimental Setup ............................................................... 94
  6.3 Evaluate the effect of varying “signal-to-noise” ratio .................. 95
    6.3.1 Primary Sequence motifs ................................................ 95
    6.3.2 Secondary structure motifs ............................................ 96
  6.4 Evaluate the effect of varying “signal-percentage” ..................... 99
    6.4.1 Primary Sequence motif ................................................ 100
  6.5 Discussion ............................................................................. 105
  6.6 Future Work .......................................................................... 107

7 Conclusion .................................................................................. 111
  7.1 Summary ................................................................................ 113
  7.2 Future Work ........................................................................... 114
  7.3 Final Remarks ........................................................................ 116
Appendix

A Preliminary Work

A.1 Introduction .......................................................... 133

A.2 Identify RBP targets: Motif Searching and Radial Basis Kernel . ................. 133
  A.2.1 Problem Description .............................................. 134
  A.2.2 Approach .......................................................... 135
  A.2.3 Results ........................................................... 136
  A.2.4 Uncompleted and Planned Directions ............................ 137
  A.2.5 Conclusion .......................................................... 137

A.3 Identify RBP targets: Kernel Methods ...................................... 138
  A.3.1 Problem Description .............................................. 139
  A.3.2 Approach .......................................................... 139
  A.3.3 Results ........................................................... 140
  A.3.4 Uncompleted and Planned Directions ............................ 142
  A.3.5 Conclusion .......................................................... 142

A.4 Identify sequence and structure of RBP binding sites: RNAPromo ............... 143
  A.4.1 Problem Description .............................................. 143
  A.4.2 Approach .......................................................... 143
  A.4.3 Results ........................................................... 144
  A.4.4 Conclusion .......................................................... 144

A.5 Background .................................................................. 144
  A.5.1 Data: Obtaining HuR binders from IP assay ....................... 144
  A.5.2 Radial Basis Kernel ............................................... 147
  A.5.3 Stem kernel, Directed Acyclic graph kernel ....................... 148
List of Tables

Table

2.1 IUPAC codes and corresponding value of bases ........................................... 22

4.1 Summary of sequence length of HuR and TTP PAR-CLIP [3] clusters .............. 48

4.2 Discriminative methods recover known K-mers for HuR. Top feature $k$-mers from
the $k$-spectrum kernel and over-represented k-mers from DREME are compared to
the published k-mers [100]. For the $k$-spectrum kernel, determined feature weights
are provided. ................................................................. 55

4.3 Discriminative methods recover known K-mers for TTP. Top feature $k$-mers from
the $k$-spectrum kernel and over-represented k-mers from DREME are compared to
the published k-mers [130]. For the $k$-spectrum kernel, determined feature weights
are provided. ................................................................. 56

4.4 Feature engineering does not consistently improve model performance. Performance
(success rate, sensitivity, PPV and ROC) for models for HuR and TTP incorporating
engineered features. ......................................................... 57

4.5 Engineered features obtain the top weights in the model. Top ten features along with
their weights discovered by the $k$-spectrum kernel method when feature engineering
is incorporated. .............................................................. 58

4.6 Four different test scenarios to identify motifs shared and specific to each RBP. .... 59
4.7 Discriminative methods evaluate distinct subsets of the HuR and TTP dataset to identify both shared and specific sequence features. The performance of the $k$-spectrum kernel on distinct subsets of the data. Top eight $k$-mers from $k$-spectrum kernel (by weight) and DREME (by E-value). The test scenarios correspond to Table 4.6.

4.8 Domain adaptation and multi-task learning identifies domain specific (prefaced with HuR or TTP) and shared (no prefix) $k$-mers. Performance metrics and top twenty $k$-mers are compared for the HuR, TTP and combined model (see main text for description of models).

5.1 Configuration Options to setup benchmarking parameters

6.1 Average and Variance of the GC-content for the created sequences

6.2 Average and Variance of the GC-content for the generated sequences

6.3 Average and Variance of the GC-content for the generated sequences

6.4 Average and Variance of the GC-content for the generated sequences

A.1 Comparison of results using motif-searching and machine-learning

A.2 $k$-spectrum kernel results. **Balanced success rate** is the fraction of correctly classified examples weighted by the size of the dataset and is used when the data sets are not equal size. **Sensitivity** is the fraction of examples from the positive class that is correctly classified. **Positive predictive value (PPV)** is the fraction of correct predictions made. **AUC** column refers to the area under the receiver operating characteristic (ROC) curve.

F.1 Comparing the motifs predicted by DREME [13] when the sequence lengths were varied with values [128, 256, 512, 1024] to the embedded k-mers for LIN-28 motif.
List of Figures

Figure

2.1 Central Dogma of Biology [194] ................................. 6

2.2 Flow of information from DNA to protein in eukaryotes. Figure is taken from Essentials of Cell Biology [2]. ................................. 6

2.3 Mechanisms of Transcript Processing ................................. 7

2.4 Different regions in RNA secondary structure. Hairpin Loop: Sequence of unpaired bases bounded by one base pair. Most common four bases long for each loop (minimum of 3 bases), and can be much longer in some cases. Bulge: The two base pairs are contiguous only at one end. Occurs when bases on one side of the structure cannot form base pairs. Interior Loop: The two base pairs are not contiguous. Occurs when bases on both ends of the structure cannot form base pairs. Junctions or Multi-loops: Any structure that is bounded by three or more base pairs. Two or more double-stranded regions converge to form a closed structure. Pseudo-knots: Any non-nested structure is referred to as pseudoknot. Stacks or Stems: Continuous nested base pairs. They are energetically favorable. Figure is taken from [3]. ................................. 8

2.5 A: RRM domains .................................................. 10

2.6 B: KH domain .................................................... 10

2.7 C: Zinc-finger domain ............................................. 10

2.8 RNA binding domains ............................................. 10
2.12 Protein-Centric Methods. **a: in vivo** approaches include native purification protocols (RNA Immunoprecipitation, RIP) and denaturing protocols (Crosslinking and Immunoprecipitation, CLIP). In the case of RIP, RNAs bound to a specific protein is immunoprecipitated from cell lysate under native conditions by using protein-specific antibody. The next step involves washing of unbound material, followed by protein removal by proteinase K treatment. The resulting RNAs are reverse transcribed and identified through RNA sequencing. In the case of CLIP, the cells are UV cross-linked to strengthen the protein-RNA complex. The RNA is digested to obtain fragments of a defined size and the obtained complexes are immunoprecipitated. After isolating the RNAs from the protein, the RNA fragments are reverse transcribed and sequenced [107]. **b: in vitro**: Step-wise procedure of RNACompete [149]. (i) RNA libraries are generated in vitro transcription. (ii) Transcripts are incubated with the protein of interest immobilized on an affinity matrix. (iii) The bound fragments are then fluorescently labeled and detected by hybridization on a microarray platform. Figure is taken from [116].

2.13 Example PWM for HuR provided by Hilal et al. [86].
3.1 Computational pipeline for PAR-CLIP analysis. Reads generated from experimental methods are processed to generate distinct reads. These reads and then aligned to the reference genome and quality filtered alignments are grouped into clusters. These clusters are scored based on various metrics and annotated against the reference genome. Appropriate cut-offs and quality scores are used to filter out false-positives and the resulting clusters are known to contain at least one binding site of the RBP being studied. This set is then subjected to statistical analysis and target peak calling. Reports are then generated in the forms of diagrams and HTML for each stage of the pipeline. Figure taken from [83].

3.2 A: Large Margin SVM
3.3 B: Soft Margin SVM
3.4 C: Kernel Mapping
3.5 SVM and Kernel Methods
3.6 $k$-spectrum kernel methods
3.7 Trie data structure example. Figures are taken from [102].
3.8 Example of a HMM model used to generate AT-rich and GC-rich sequences[37].

4.1 **Feature Augmentation Technique:** Original features and augmented features for HuR and TTP domains

4.2 Position Frequency Matrix for the top motif identified by DREME for HuR
4.3 Position Frequency Matrix for the top motif identified by DREME for TTP

4.4 **Analysis of Feature** $k$-mers: (A) Number of occurrences of HuR specific feature $k$-mers in each dataset. (B) Number of occurrences of TTP specific $k$-mers in each dataset. (C) Number of occurrences of common $k$-mers in each dataset.

5.1 A: Single-stranded RNA
5.2 B: Vts1p binding to single stranded region of secondary structure
5.3 C: Double-stranded binding
5.4 Examples of RBPs binding to different domains ........................................ 67
5.5 Sequence alignment of different RRM domains. Conserved RNP1 and RNP2 se-
quences are displayed in yellow. Figure taken from [117] ............................... 69
5.6 TTP bound to the TZF domain identifying the 9-mer sequence UUAUUUAUUU.
Figure taken from [32] ................................................................. 70
5.7 dsRBD: Comparison of dsRNA and double-stranded DNA helix. Figure is taken
from [119] ................................................................................. 71
5.8 Sequence Generator WorkFlow ................................................................. 72
5.9 Process of creating Positive and Control set of sequences ......................... 76
5.10 Comparison of 3-grams of 3’ UTR and 5’ UTR transcripts and created positive and
negative sequences. ................................................................................. 79
5.11 Overview of evaluating motifs identified by DREME [13] .............................. 80
5.12 Scenario 1: Computing True Positive, False Positive, False Negative ........ 82
5.13 In this case, the predicted $k$-mer is found after the end of the embedded $k$-mer.
Thus, the entire length of the embedded $k$-mer is the number of false negatives or
$falseNegatives = embeddedEnd - embeddedStart$. $falsePositives = predictedEnd -$
$predictedStart$. $truePositives = 0$ as the computational tool did not find any motifs
at the location where the signal was embedded. ........................................... 82
5.14 The figure illustrates the simplicity of the framework and the lines of code map
to the Sequence Generator architecture in Figure 5.8. Adding a new signal type
will require changes to the function $CreateConfFiles()$ which generates the individ-
ual experimental configuration files and the function $GenerateFastaFile()$ which will
embed the new signal type in the FASTA files. Adding a new computational method
to the framework will require changes to functions $RunComputationalTools()$ and
$ParseResultsAndGenerateGraphs()$. See the next figure for more details. .......... 84
5.15 The figure illustrates the RunComputationalTools method that iterates through the directory looking for Signal and NoSignal files, and sequentially runs through the computational tools. DREME and $k$-spectrum are examples shown. This function will require an addition of a function call to add a new computational method to the framework. The individual method will then invoke the computational tool passing in the right parameters.

5.16 Components involved in the binding of RBPs to target mRNA

5.17 Generation of positive sequences: Improved process to include RNA binding domain followed by embedding the motif

6.1 Sequence logos of sites bound by RNA binding proteins that recognize primary sequence. These four RBPs were benchmarked, and their motifs were obtained from CIS-BP-RNA

6.2 Sequence logos of RNA structure elements. Figures for each structure is obtained from Rfam DB

6.3 Primary Sequence Motif: RBP LIN-28: Effect of varying the signal-to-noise ratio. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

6.4 Secondary Structure Motif: RNA structure element: LIN-4: Effect of varying the signal-to-noise ratio. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

6.5 Primary Sequence Motif: RBP LIN-28: Effect of varying the signal-percentage. The number of sequences that contain the embedded signal is varied between 10 percent and 100 percent. The sensitivity and PPV values are computed at every level of signal percentage.
6.6 Secondary Structure Motif: RNA structure element: LIN-4: Effect of varying the signal-percentage. The number of sequences that contain the embedded signal is varied between 10 percent and 100 percent. The sensitivity and PPV values are computed at every level of signal percentage.

6.7 Primary Sequence Motif: RBP LIN-28: Effect of varying the signal-to-noise ratio. Results obtained by using Sequence-based evaluation method. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

A.1 The End-to-End Process of Predicting HuR Binders
A.2 Length of 3’UTR sequences
A.3 HuR motifs found by RNAPromo in 3’UTR region
A.4 Acquisition of Potential HuR Binding mRNAs. Figure created in collaboration with Dr. Nick Farina.
A.5 Two-dimensional version of such a RBF kernel
A.6 Averaged base-pairing probability matrices and DAG kernels using dynamic programming used to calculate profile-profile stem kernels for multiple alignments of RNA sequences. (a) Given multiple alignments, (b) Base-pairing probability matrices for each sequence in the multiple alignments and average base-pairing probabilities with respect to the columns of each alignment. (c) Build a DAG with averaged base-pairing probability matrix, where each vertex corresponds to a base pair whose probability is above a predefined threshold, (d) Kernel value for a pair of DAGs for multiple alignments by using DAG kernel and the dynamic programming technique.

D.1 Evaluate the effect of increasing sequence length on sensitivity and PPV for RBPs HuR, TTP, and FUS
D.2 Evaluate the effect of increasing the number of sequences that contain the embedded signal on sensitivity and PPV for RBPs HuR, TTP, and FUS ............. 155

D.3 Evaluate the effect of increasing the signal to noise ratio that contain the embedded signal of miRNA mir-101, rna structure element IRE on sensitivity and PPV ....... 155

D.4 Evaluate the effect of increasing the number of sequences that contain the embedded signal of miRNA mir-101, rna structure element IRE on sensitivity and PPV ....... 156

E.1 Review of existing Sequence Simulators [137] .................................................. 158

G.1 Comparison of HuR and TTP transcripts: HuR is known to bind to larger 3´UTR sequences where as TTP is known to bind to shorter 3´UTR transcripts. However, they are also known to be competitive and share binding sites .......... 163

H.1 Adapted from Lope´z de Silanes et al.[41]. (A) A probabilistic matrix of the predicted HuR motif indicates that the motif is U-rich. (C) Gives examples of representative secondary structures as predicted. We used the probabilistic matrix and the secondary structure to build the descriptor for the RNAMotif program [114] ....... 164
While most people know at some level that genes in Deoxyribonucleic acid (DNA) are transcribed into Ribonucleic acid (RNA) and then translated into proteins, many are unaware of the numerous mechanisms that help regulate these processes. For example, once an RNA molecule is created, it may be rapidly degraded so that no protein is made from it (for example, if there is currently too much of that protein), or, if the protein product is urgently needed, the RNA molecule may be shielded from such degradation. Some proteins affect this process by binding directly to the RNA. An improved understanding of how these RNA-binding proteins (RBPs) bind to RNA would help us understand the mechanisms that govern the post-transcriptional processes.

Gene expression is the process by which information contained in a gene is read to create a genetic product (either proteins or RNA). Regulation of gene expression is necessary to decide what product needs to be created and in what amount. Transcription factors bind to regions of the DNA and regulate nearby transcription of genes. Post-transcriptional regulation is the control at the RNA level and what happens to it between transcription and translation of the gene. RNA-binding proteins (RBPs) are crucial regulators of numerous post-transcriptional processes such as splicing, stability, degradation of target messenger RNA (mRNAs).

The RBPs contain one or more RNA-binding domains (RBDs) and they recognize their targets in highly specific fashion, either through recognition of specific primary sequence or secondary structure. RBP binding alters the stability of the bound mRNA and thus controls gene expression. The importance of RBPs is further pronounced by their impact on several genetic diseases,
tumor development [200], as well as their role in several cancers [1, 24, 152, 87]. Human antigen R (HuR) and Tristetraprolin (TTP) are two RBPs that exhibit opposing effects when bound to mRNA transcripts but are known to have similar binding preferences. While HuR protects the bound mRNA from degradation, binding to Tristetraprolin (TTP) results in the decay of the target RNA. In addition, there also exists evidence of competitive binding between these RBPs where TTP is displaced with the increasing amount of HuR [180]. However, the binding sites that are shared by the RBPs and those that are specific to each RBP are unclear. Support Vector Machines (SVMs) have been used successfully in predicting RNA-binding proteins from their primary sequence [35, 73, 166], but have not been explored as a method to identify targets of these RBPs i.e. the mRNAs where the RBPs bind to. Hence, I explore the use of SVMs in conjunction with kernel methods to identify mRNA targets of RBPs.

1.1 Thesis Outline

In this thesis, I evaluate the existing \(k\)-spectrum kernel method as an approach to identifying binding characteristics of two well-studied RBPs (HuR, TTP) that are known to bind to primary sequence. In order to identify specific and shared binding sites of HuR and TTP known to have similar binding preferences, I extend the kernel method to incorporate domain adaptation. In order to evaluate the performance of computational methods under different biological conditions (e.g. variations in sequence lengths, binding preferences of the RBP being studied, etc), I developed a framework in Python called the “PySG” that would benchmark the performance of computational methods under varying biological conditions. This framework generates sequences by embedding particular motifs (primary and secondary structure-based), varies the number of sequences that contain the motif, varies the noise present in sequences containing the motif, and can also simulate other biologically relevant experiments. The framework then benchmarks the performance of computational methods on the generated set of sequences.

The thesis is organized into chapters as follows:
• Chapter 2 introduces relevant biology concepts to this thesis. The background material introduces RNA, the process of creating RNAs, understanding the post-transcriptional regulation process, and the role RBPs play in that process. Experimental methods, as well as existing computational methods used to identify RBP targets, are also reviewed in this chapter.

• Chapter 3 reviews the motivation of selecting SVM and kernel methods as an approach to identify RBP targets. Basic concepts of SVMs, the $k$-spectrum kernel method, as well as domain adaptation are reviewed in this chapter.

• Chapter 4 illustrates the use of the $k$-spectrum kernel method to predict the targets of RBPs: HuR and TTP. Secondly, domain adaptation is explored as a technique to identify binding sites specific to each RBP, as well as those shared by them. The content of this chapter is published at the PLOS One journal titled “Discriminating between HuR and TTP binding sites using the $k$-spectrum kernel method”.

• Chapter 5 explains the architecture of the PySG framework that I developed that allows bioinformaticians to generate custom datasets by embedding either primary sequence-based motifs or secondary structure alignments followed by benchmarking computational methods on the generated sequences. The framework is available publicly on GitHub.¹

• Chapter 6 demonstrates the use of the PySG framework and benchmarks two discriminative primary sequence-based computational methods against the datasets generated under the experimental conditions specified in the chapter.

• Chapter 7 summarizes findings of this thesis, enumerates the contribution to the field of bioinformatics, and finally provides future directions for this work.

• Appendix A describes the preliminary research I carried out to identify RBP targets. This chapter is provided for completeness of the thesis and is focused on using transcript se-

¹https://github.com/shwetabhandare/PySG/
quences (as opposed to regions where the RBP was bound) to identify genome-wide targets of RBPs. A combination of motif-searching and machine-learning approach is also discussed. The understandings gained from preliminary research is listed and guided the research carried out for this thesis. The changes in direction carried out after this preliminary work are explained by the availability of newer experimental methods, as well as the failure of current computational methods to execute on longer transcript sequences.
Chapter 2

Biology Background

This chapter reviews the concepts required to comprehend the biological aspects that are related to the identification of RNA-binding proteins (RBPs). Section 2.1 provides a basic understanding how RNA is created. Section 2.2 explains the role of RBPs and the mechanisms by which they recognize their targets. Different experimental methods used to identify RBP targets are discussed in Section 2.3. The challenges of these experimental methods are listed in Section 2.4. The chapter concludes by reviewing the existing computational methods for analyzing RBP targets in Section 2.5.

2.1 Brief RNA Primer

RNA, like Deoxyribonucleic acid (DNA), is made up of a long chain of components called nucleotides. The primary structure of RNA is a sequence that is made up of four nucleotides {A, C, G, U}. There are a number of distinct chemical properties to RNA as compared to DNA such the presence of ribose sugar backbone that has a hydroxyl (-OH) group in RNA, and its absence in DNA. Secondly, DNA is usually found in a double-stranded form whereas RNA is typically found in a single-stranded form. Also, RNA uses uracil (U) as its base as compared to thymine (T) for DNA [53]. RNA plays an important role in the process of converting DNA to proteins, also known as “Central Dogma” (See Figure 2.1) of molecular biology which states that “DNA is used to produce RNA which is subsequently used to produce proteins” [5].

The process of creating RNA involves reading a DNA template and producing RNA copy of
the required parts of its DNA strand that contains coding information (gene). This process is called **transcription** and the generated RNA strand is identical to the DNA strand except for the base U instead of T. This strand of RNA may then be **translated** to create a protein. Messenger RNA (mRNA) is transcribed for regions of DNA coding for protein and provides a chemical “blueprint” for a protein product. Figure 2.2 outlines the process that takes place in the nucleus of the cell. It is important to note that the compartmentalization, i.e. nucleus and cytoplasm, is distinctly eukaryotic. As seen from the Figure 2.2, the DNA strand is always present in the nucleus of the cell, whereas the generated mRNA travels into the cytoplasm.

**Transcript Processing:** The process of transcript maturation adds stability of the message. The generated RNA strand contains both the exons (coding sections) and introns (non-coding sections). This strand is spliced to remove the intronic regions, and the exons are merged together to create an mRNA strand. This mRNA strand is further modified to add a 5´cap, and 3´poly-A tail. Capping at the 5´ end protects the mRNA from degradation, promotes ribosome binding, and
regulates nuclear export. The process of polyadenylation is the addition of multiple adenine (A) bases to the mRNA strand also known as the poly-A tail. This acts as a buffer for exo-nucleases and helps to increase the half-life of the mRNA. Thus, it serves a similar purpose as the capping, and protects the mRNA from degradation, promotes translation, and regular nuclear export. It also helps with transcription termination for the RNA polymerase that is transcribing the messenger RNA. The poly-A tail is variable in length but on average about 250 nucleotides long. Summarizing, the capping and poly-A tail addition help to stabilize the mRNA for translation (See Figure 2.3).

**Figure 2.3:** While transcriptional regulation of genes greatly affects expression, many mRNAs undergo further transcript processing resulting in differential protein expression. Alternative splicing of the pre-mRNA leads to translation of functionally unique proteins. AU-Rich Element (ARE) binding proteins bind to AU-rich sequences and affect transcript stability through recruitment or inhibition of decapping enzymes and deadenylases. The miRNA: RISC (miRISC) complexes bind through the imperfect base pairing of the miRNA to the 3′-UTR of target mRNAs and lead to translational inhibition or mRNA decay. Poly-A binding proteins bind to the poly-A tail at the 3′-UTR of mRNAs and stabilize the mRNA by protecting the mRNA from RNases. Figure created in collaboration with Dr. Nick Farina.

While the mRNA transcript generated is single stranded in nature, RNAs can fold intramolecularly to form a number of short base-paired structures called secondary structure. The folding of an RNA sequence is governed by the formation of intramolecular pairs. Base pairing in RNA sequences contributes to the stability of the RNA structure. For RNAs where the structure is functionally important (such as tRNAs), the changes in the sequence are such that the base-pairing is maintained. This process is known as covariation which ensures the ability to base pair is maintained and RNA structure is conserved. In RNA sequence analysis, the specific form of the secondary structures in the cell is an important feature for modeling and detecting RNA sequences. Figure
Figure 2.4: Different regions in RNA secondary structure. **Hairpin Loop:** Sequence of unpaired bases bounded by one base pair. Most common four bases long for each loop (minimum of 3 bases), and can be much longer in some cases. **Bulge:** The two base pairs are contiguous only at one end. Occurs when bases on one side of the structure cannot form base pairs. **Interior Loop:** The two base pairs are not contiguous. Occurs when bases on both ends of the structure cannot form base pairs. **Junctions or Multi-loops:** Any structure that is bounded by three or more base pairs. Two or more double-stranded regions converge to form a closed structure. **Pseudo-knots:** Any non-nested structure is referred to as pseudoknot. **Stacks or Stem:** Continuous nested base pairs. They are energetically favorable. Figure is taken from [2].

2.2 RNA binding proteins and their roles in regulation

RBP s are known to bind RNAs during transcript processing. These mRNA untranslated regions serve as binding domains for regulatory proteins. The binding to this region has an impact on the stability and localization of the resulting transcript. There are several known RBPs each having their own function [67]. For example, HuR is an RNA binding protein that primarily binds to AU-Rich elements (AREs) in the 3’ UTR regions of the mRNA and stabilizes the transcripts by protecting them from degradation in the cytoplasm. While it is unclear whether the presence of AU-rich elements in a transcript constitutes to the stability of the transcript, it is known that these regions are known to harbor binding of RBPs which then impact the stability of the bound transcript [25, 139]. Poly-A binding proteins (PABPs) are known to bind to the poly-A tail at the 3’ UTR end and also stabilize the mRNA from exo-nucleases chewing up the transcript. TTP is a zinc-finger RBP which is also known to bind to AREs in the 3’ UTR regions of the mRNA, but binding of TTP promotes the degradation of the target transcript. While binding to some RBPs
stabilizes the resulting transcript, others result in degradation of the target mRNA. RBPs are also known to play an important role in alternate splicing. Mutations in RBPs, variations in RNA processing are known to result in several neurodegenerative disorders, fragile X syndrome, muscular atrophies, human tumors, as well as cancer \cite{60}. RBPs regulate cell growth and proliferation, and dysfunction in this area can lead to cancer. Several types of cancer in humans are linked to RBP defects such as lung cancer, breast cancer, chronic myeloid leukemia to name a few \cite{87, 151, 200}. Thus, it becomes important to understand mechanisms by which RBPs bind to RNA which is discussed in the next section.

### 2.2.1 RNA binding domains

The structure of the RBPs and the RNA-binding domains (RBDs) enables them to bind to RNAs. RBDs either recognize single-stranded RNA (ssRNA) or double-stranded RNA (dsRNA) when bound to the target RNA. Motifs are short patterns that exist in DNA or RNA sequences and have biological significance. Sequence-specific motifs are typical in binding sites for proteins such as transcription factors (TFs). For RNAs, motifs are patterns that could exist in primary sequence as well as the secondary structure and are involved in ribosome binding, mRNA processing, and transcription termination. There are 40 different types of RBDs that are known to be involved in RNA binding \cite{93}, some of the domains include RNA recognition Motif (RRM), KH (K-homology), CSD (cold-shock domain), Zinc-finger, double-stranded RBB (dsRBD), Arginine-rich, RNA helicase, PAZ, and PIWI \cite{93}. The length of RNA binding protein target sites is typically between 4 – 40 nucleotides long and is dependent on RBDs that are involved in the binding \cite{117}. ssRBDs such as RRMs, KH domains, zinc-finger domains typically recognize degenerate 4 – 6 nucleotide regions and are known to occur in combinations or as repeats thus resulting in increased sequence specificity and affinity of RBPs \cite{10, 112}. Since the domains RRM, K-Homology, and Zinc-finger are the most prevalent domains, they will be reviewed in the next paragraphs (See Figure 2.8).

**RNA recognition Motif (RRM):** is one domain that is prevalent in eukaryotes and bacteria. In bacteria, only 85 proteins containing an RRM domain have been discovered so far when compared
to 6,056 RRM motifs have been identified in 3,541 different proteins in eukaryotes [112]. Bacteria RRM proteins are smaller and contain only a single copy of the RRM domain. In humans, around 500 proteins containing at least one RRM domain has been discovered, accounting for almost 2% of the gene products. In contrast with bacteria RRMs, eukaryotic RRMs are often found as multiple copies within a protein [117] (2 – 6 copies) and/or associated with other domains. Heterogeneous nuclear ribonucleoprotein (hnRNP) A1, polypyrimidine tract binding (PTB) protein, SF2, and Hu proteins are examples of proteins that contain RRM as their binding domain.

PTB proteins are known to bind to pyrimidine-rich tracts in the 3’ end of introns and are known to suppress splicing. PTB is known to contain four RRMs, whereas SF2 contains two RRM domains. PTB is primarily known for its role in alternative splicing but it is also involved in regulation of translation. The Hu proteins (HuR, HuD) are known to bind to AU-rich elements (AREs) that are generally present in the 3’ UTR regions of the mRNA affecting the stability of the bound transcript. These untranslated regions in part serve as binding domains for regulatory proteins. The binding to this region has an impact on the stability of the resulting transcript.

KH domain: RBP s that contain the KH domain are known to recognize both ssRNA and dsRNA
and are involved in a wide range of processes such as translation, splicing, and transcription. This domain is present in eukaryotes (eubacteria and archaea) and is about 70 amino acids long. There are two types of KH domains: type I and type II. Type I is mostly found in eukaryotic proteins whereas type II is found in bacteria proteins. The KH domains bind a target of four nucleotides and the binding is achieved through hydrogen bonding, electrostatic interactions, and structure reciprocity. An example of RBPs in this domain includes hnRNP K which includes three KH domains (KH1, KH2, KH3) and is known to directly interact with signal transduction machinery which affects gene expression at the post-transcriptional level [184].

Zinc-finger: Proteins that contain this domain consist of one or more zinc ions that are necessary to stabilize the folding. The binding properties of this domain depend on the number of zinc fingers, the amino acid composition of the finger domains as well as on the linker between fingers. The motif associated with this domain can be found by itself, or as a repeated domain or possibly with other RBDs. The domains are classified based on the amino acids that interact with the zinc ion (CCHH, CCCH, or CCCC). They were first identified as a DNA-binding motif in transcription factor TFIIIA. DNA recognition of the TFIIIA domain involves sequence whereas RNA recognition is dependent on the three-dimensional structure. TIS11d is a member of the Tristetraprolin (TTP) family and is known to contain two CCCH zinc fingers. Similar to HuR, TTP is known to bind to the AU-rich elements and the binding is known to degrade the mRNA transcript [58].

Conservation of RBPs

RBDs are highly conserved across bacteria, archaea, and eukaryotes [65]. The composition of individual domains in a protein is functionally important as well as the manner in which the domains are arranged relative to one another [112]. Domains that occupy the same position in orthologous proteins are known to exhibit higher levels of conservation when compared to domains in the same protein but a different position. One such example is the RRM1 domain in yeast is similar to the RRM1 domain of the human protein than it is to the RRM3 or RRM4 domains of the yeast protein [112]. Gerstberger et al. [65] studied the evolutionary conservation across species of 1,542 RBPs that interact with all the classes of RNAs. They grouped these RBPs into categories
based on the type of targets that they were bound to such as mRNAs, tRNAs, rRNAs (ribosomal RNA). The RBPs studied were involved in functions such as protein synthesis, gene regulation, gene expression control; these functions are essential for cellular life, and they concluded that the RBPs were highly conserved. Figure 2.11 [112] illustrates the conservation of these RBPs families. Giang et al. [84] identified five out of a set of fifteen RBPs that were highly conserved in the 3’ region, the RBP and the conserved motif were: PUM(UGUANAUA), Fox-1 ((UGCAUGU), U1A (AUUGCAC), Nova (YCAUUUCAY), and the AU-rich element (ARE) UAUUUAU. The final ARE motif is bound by different RBPs [16, 27]. Maruyama et al. [118] studied the conservation of structure of RBP in cyanobacteria for the RRM domains. They compared the 5´ UTR of all known RNA binding protein “rbp” genes of cyanobacteria and found two conserved regions located around the translation initiation codon. The first was the ribosome-binding site located about six bases upstream and the second was the motif 5´-UCUCCGAA-3´ located about 40 bases upstream of the initiation codon. The latter motif was highly conserved in all of the cold-regulated rbp gene sequences that were analyzed. Maruyama et al. [118] also studied the relationship between bacteria Rbp proteins and eukaryotic RNA-binding proteins of type RRM by construction an alignment and phylogenetic tree of 30 representative RRMs of various organisms. They found the amino acids within the motifs RNP-1 and RNP-2 are highly conserved and found some residues are conserved in almost all RRMs (See Figure 6 in [118] for more details).

The RBP binding involves both primary sequence and structure as seen the review of different RBDs [116]. Experimentally methods used to identify RBP targets are outlined in the next section.

2.3 Experimental methods to identify target RBPs

Experimental methods can be classified into in vivo and in vitro. in vitro in Latin refers to “within the glass (controlled environment)” whereas in vivo in Latin means “within the living (cell),” are two different types of methods used to detect RBP targets. Advances in sequencing techniques have led to the development of high-throughput methods which simultaneously detect thousands of interactions in a single experiment. The experimental methods are classified into
protein-centric and RNA-centric approaches [116]. In protein-centric approach, the RNAs bound to the protein are identified whereas in RNA-centric approach, the proteins interacting with the RNA of interest are identified. This thesis is focused on protein-centric approaches, and the next section will review both in vivo and in vitro methods and discuss the pros and cons of each approach.

2.3.1 In vitro Methods

Selection of ligands by exponential enrichment (SELEX): is a low-throughput method for in vitro identification of RBP targets [52]. In this process, a large pool of single-stranded RNA or DNA sequences 15 to 60 bases in length are exposed to a target protein or antibody. The sequences that have high affinity to the protein are selected through multiple rounds of selection, purification, and amplification. The resulting bound RNAs are cloned and sequenced thus generating a set of high-affinity targets. SELEX is known to have a bias towards high-affinity targets and does not identify sequences that have lower specificity or medium-affinities to single protein or protein complexes.
Figure 2.12: Protein-Centric Methods. **a: in vivo** approaches include native purification protocols (RNA Immunoprecipitation, RIP) and denaturing protocols (Crosslinking and Immunoprecipitation, CLIP). In the case of RIP, RNAs bound to a specific protein is immunoprecipitated from cell lysate under native conditions by using protein-specific antibody. The next step involves washing of unbound material, followed by protein removal by proteinase K treatment. The resulting RNAs are reverse transcribed and identified through RNA sequencing. In the case of CLIP, the cells are UV cross-linked to strengthen the protein-RNA complex. The RNA is digested to obtain fragments of a defined size and the obtained complexes are immunoprecipitated. After isolating the RNAs from the protein, the RNA fragments are reverse transcribed and sequenced [107]. **b: in vitro**: Step-wise procedure of RNACompete [149]. (i) RNA libraries are generated in vitro transcription. (ii) Transcripts are incubated with the protein of interest immobilized on an affinity matrix. (iii) The bound fragments are then fluorescently labeled and detected by hybridization on a microarray platform. Figure is taken from [116].

**RNACompete**: is a high-throughput in vitro method that estimates relative binding affinities
of selected RBPs to each RNA sequence in a defined population [149]. RNACompete [149] consists of three basic steps as shown in Figure 2.12: (1) **RNA Library generation**: an RNA pool comprising each \( k \)-mer in a variety of structural contexts, (2) **RNA pulldown**: a single pulldown of the RNAs bound to a tagged RBP of interest, and (3) **Microarray analysis**: of the relative enrichment of each RNA in the bound fraction relative to the initial pool.

When there are multiple proteins present in the pool of RNA sequences, the RNACompete [149] method as the name suggests harbors a competition during the pulldown phase. This method identifies the RNA sequences bound to the proteins and would depend on the relative affinity of the RNAs to the protein. Thus, the ratio of RNA in the pulldown to the RNA population in the pool provides a measure of binding preference for the protein.

While both these methods are in-vitro, SELEX [52] identifies the high-affinity targets which do not always reflect the physiological binding sites. This results in missing out on the protein's affinity to the sub-optimal motifs [52]. One short-coming of **in vitro** experimental methods is their inability to duplicate the conditions present inside the cell when the experiment was carried out.

### 2.3.2 **In vivo Methods**

**In vivo** methods capture direct biological conditions happening in the cell and hence are more physiologically relevant. However, they do require the RBP to be active in the cell during the experiment. Figure 2.12 (a) illustrates the two examples described below.

**RNA immunoprecipitation (RIP)**: is an example of a high-throughput **in vivo** experimental method. RIP purifies RBP-mRNA complexes from cellular extracts and identifies mRNAs bound to a protein-specific antibody followed by using either a microarray (RIP-chip) or high-throughput sequencing (RIP-seq) [89 206]. It is possible that the targets might become dissociated from the protein as there is no cross-linking of the RBP to the mRNA. It is also possible to capture non-specific binding of the RNAs due to the other high-affinity targets during cell lysis [124]. This technique specifies the transcript to which the RBP is bound and not the region within the transcript where the binding occurs.
Cross linking and immunoprecipitation (CLIP or High-throughput sequencing-CLIP (HITS-CLIP)): eliminates the short-coming of RIP by incorporating ultraviolet (UV) cross-linking of the RBPs to the RNA before immunoprecipitation (See Figure 2.12 (a). The cross-linking step is followed by stringent purification of protein-RNA complexes before proceeding to the high-throughput sequencing of cDNA library. Cross-linking has several benefits: the number of false positive targets are reduced as the strong bonds introduced by cross-linking permits for a more stringent washing procedure. Furthermore, cross-linking is also known to protect the target site from ribonuclease digestion resulting in a much greater resolution in identifying the actual binding site.

Photoactivable ribonucleoside-enhanced CLIP (PAR-CLIP): technique incorporates ribonucleoside analogs that are photoreactive, such as 4-thiouridine (4-SU) and 6-thioguanosine (6-SG) into nascent RNA transcripts in living cells. Using the CLIP experimental method, it is hard to precisely identify the location of the crosslink within the sequenced cross-linked fragments. This makes it hard to separate the target RNA segments from background non-crosslinked RNA fragments. The cross-linking in PAR-CLIP involves the addition of a characteristic mutation in the sequence (T to C conversion). This allows for the precise binding sites of the RBP to be detected. This modification to the original CLIP protocol has thus increased the resolution of the binding sites. The RNA isolated from this process is then converted to a cDNA library followed by deep sequencing using next-generation sequencing technology which results in sequences that are bound to the RBP. This method should be favored when a higher resolution of binding sites is necessary as this method has the highest number of characteristic mutations on cross-linked sites compared to the other three.

Individual-nucleotide CLIP (iCLIP): Another modification to the CLIP protocol iCLIP allows single-nucleotide resolution of RBP targets. In this technique, after the cross-linking steps, adapter sequences are joined to the 3´ ends of the RNAs. The bound protein is then partially digested with proteinase K thus leaving a polypeptide at the binding site which provides the exact location of the binding site. This allows single nucleotide resolution as the cDNA synthesis knows
the exact start location (adapter position) and the end of the site (polypeptide). However, among the three methods, this is technically the most challenging as in order to get the right reads, the RBP bound to the mRNA needs to be partially digested which requires being aware of the conditions for the digestion. The digestion ensures that the reads are long enough to be aligned.

2.4 Challenges of analysis of experimental data

All the experimental methods discussed in the previous section produce sequences from which the goal is to learn the recognition signal (primary or secondary). This section will discuss the limitations of the techniques in recognizing the signal.

The growing use of high-throughput PAR-CLIP [72] experiments has led to the availability of large-scale datasets of target mRNAs with binding sites for individual RBPs. However, these methods have limitations such as identifying false positive targets which are the result of contamination with non-cross-linked sites. For high-throughput experiments, there are three common types of errors: insertions or deletions of a base in a read (indel errors), base miscalled in a read (substitution errors), and finally based on the sequencing platform, the error profile associated with the instrument. Illumina and Roche/454 are two main sequencing instruments. Illumina sequencers have a low probability of indel errors, and hence substitutions are more prevalent more so towards the end of the read [109, 144] whereas the Roche platform the complete opposite: the indel errors are ten times more than the substitution errors and the error rate does increase towards the end of the read [144]. The PAR-CLIP [72] protocol during the cross-linking stage can also introduce specific biases as it introduces T to C mutations in the reads. The CLIP techniques identify the binding sites that are in the range of 100-200 nucleotide resolution whereas RIP provides transcript-level resolution and hence do not pinpoint the exact binding location. The experimental methods require sequencing as the final step which can introduce errors due to miscalling of one or more bases thus leading to inaccuracies in the generated reads.

Finally, it is impractical to perform a functional assay for every uncharacterized protein. The in vivo methods also rely on the natural presence of the mRNA targets, but the expression of
these targets vary by cell type and condition. These limitations make the task of determining RBP targets challenging, and in the next section, computational methods developed to overcome these challenges are discussed.

2.5 Review of Computational tools for RBP analysis

Recently, Morris et al. [105] have published a review on computational and experimental methods in detecting RBP targets. In this section, I cover the highlights of that review. DNA-binding proteins recognize short primary sequence motifs for their binding whereas RBPs are known to bind to either single stranded or double stranded regions. In addition, most RBPs are known to recognize single-stranded regions or the single-stranded parts of the double-stranded structure [45]. Additionally, since this thesis is focused on discriminative methods, the Discriminative Regular Expression Motif Elicitation (DREME) computational tools will be reviewed in detail.

2.5.1 Motif-Searching Computational Tools

Because much binding recognition involves primary sequence motif discovery, prior methods used to identify RBP targets were focused on DNA-binding motifs [64, 170, 198, 207]. Here, I describe a small number of approaches that are commonly used on RNA.

Multiple EM for Motif Elicitation (MEME) [12] is a widely-used motif discovery tool that identifies un-gapped overrepresented motifs in a given set of sequences. MEME takes as input one or more sequences that are known to contain one or more motifs and outputs a series of motif models. Motifs in MEME [12] are position dependent and represented using position-specific letter-probability matrices that describe the probability of each nucleotide at each position. The motifs do not contain gaps and patterns with gaps are separated by MEME into individual motifs. MEME carries out a search for motifs based on the width specified by the user, and the algorithm generates a model for each width.

The search for the best motif is done in two phases. In the first phase, MEME iterates through the models and selects the best motif using an objective function based on the statistical
significance of the log likelihood ratio (LLR) of the occurrences of the motif. Each motif is assigned an E-value which is computed by finding the number of motifs that would have equal or higher log likelihood ratio if the input sequences had been generated randomly according to the background model. The statistical significance of the motif is computed based on the LLR, width of the motif, the number of occurrences, type of the motif model, and the size of the input dataset. The LLR value of a motif is a measure of how different the motif is compared to its background model. In the second phase of the search, MEME uses Expectation-Maximization (EM) to search for motifs based on a motif model of a particular fixed width and uses an initial estimate on the number of times the matches. This is followed by sorting through the possible matches using probability according to EM. E-values of the first \( n \) sites is computed and a sorted list of different values of \( n \) is generated. This process (EM, followed by computation of E-values) is repeated with different motif lengths and different initial estimates of the number of matches. The motif with the lowest value is output by MEME.

Amadeus [108] is another example of a computational method that is used for both DNA and RNA sequences. This method detects short motifs that are over-represented in the 3’ UTRs or promoter regions as compared to a large background set or the entire genome. The tool starts with a set of all possible \( k \)-mers, and biologically relevant filters such as HG enrichment score, binding site localization, strand bias, and chromosomal preference to converge on the final set of motifs.

Another de-novo motif searching tool is Conserve evidence-ranked motif identification tool (cERMIT) [61]. This tool uses rank ordering to search for sequence motifs based on experimental evidence (such as genome-wide binding data). cERMIT [61] identifies sequence motif represented as a \( k \)-mer over the alphabet of the International Union of Pure and Applied Chemistry (IUPAC symbols) and uses an enrichment function to score these motifs. ceRMIT [61] ranks all the possible target regions based on known binding characteristics and identifies sequence motifs that are highly enriched in these binding regions.

Finding informative regulatory elements (FIRE) [50] is another computational method that detects both DNA and RNA motifs from gene expression data associated with the flanking region
of the gene. FIRE [50] first considers all possible 7-mers, and each 7-mer is scored based on the presence/absence profile which is the prior knowledge of whether such a motif is present or absent within the regulatory region. The 7-mers are sorted based on the scores and are then optimized into motif representation. Only certain positions in the motif are allowed to change, thus identifying continuous expression profile of over-represented motifs. This technique is applied to RNA motifs as well; motifs found in the 3’ UTR regions are considered for the post-transcriptional role.

2.5.1.1 Discriminative Regular Expression Motif Elicitation (DREME)

DREME [13] is a discriminative motif searching tool that identifies enriched short, ungapped motifs in the positive set compared to the control set. The input to DREME [13] is a set of positive sequences and an optional set of negative sequences. DREME [13] generates a negative set, if not specified by the user, by shuffling the nucleotides of sequences in the positive set maintaining the dinucleotide distribution. Negative set sequences should ideally have the same length distribution as the positive set to ensure that the classifier does not use the length of the sequences as a discriminative feature. DREME uses Fisher’s Exact test to determine the significance of each motif found in the positive set as compared to its representation in the negative set. This test computes the probability that the fraction of sequences in the positive set matching the motif would be greater than that of the negative set. Only the sequences that contain a motif are counted in both the datasets. A position-specific probability matrix [174] is created once the motif with the highest significance is found. The process of finding the motif with the highest significance is continued until no new motif with E-value less than a significance threshold is found. To identify the next significant motifs in the sequences, the first motif found is erased from the sequences and the same search procedure is repeated.

By default, DREME computes the significance of each word of length three to eight nucleotides, however, this it allows configurable minimum and maximum motif lengths that control the width of the motif. DREME [13] first determines all possible sub-sequences that match the minimum and maximum motif size. Next, it computes the number of sequences that contain these
motifs for both the positive and negative datasets. The Fisher’s exact test is then applied on the motifs found to get the counts of sequences that contain words in the positive set out of the total sequence count in the positive set and the counts of sequences that contain words in the negative set out of the total sequence count in the negative set. The most significant sub-sequences are then combined into IUPAC regular expression to check if there is an improvement in the significance of the sub-sequences. Finally, to create the Position Weight Matrix, the IUPAC regular expression is then used to identify all matching sites in the positive set. These matches are combined to create a motif.

DREME has been used to identify transcription factors for eukaryotes as well as cofactors in mouse ES cells, mouse erythrocytes, and human cell line ChIP-seq datasets. DREME [13] scales linearly identifies discriminative motifs and also provides statistical significance of the discovered motifs. DREME [13] is a part of the well-known and extensively used motif searching tool MEME (Multiple EM for Motif Elicitation) [12].

DREME Output interpretation: An example output of DREME [13] is illustrated in Appendix B. DREME [13] first outputs the command that was used to run DREME [13], the size of the positive and negative dataset, the version of MEME [12], background letter (ATGC) frequency, and finally information about the predicted motif. The predicted motif is displayed in IUPAC codes illustrated in Table 2.1. The best motif is highlighted along with its occurrence in the positive and negative sets, associated P-values, and E-values.

P-value: helps determine the significance of results. When the null hypothesis is true, P-value is the probability of finding the observed results in a single random sequence for a given background model of nucleotide distribution (ATGC). So, in our case, this is the significance of the predicted motif in the positive set. A large p-value (> 0.05) indicates weak evidence against the null hypothesis which means, the predicted motif isn’t significant and that can be rejected. The 0.05 value for p-value interpretation is arbitrary and just “by convention” and requires multiple hypothesis testing. p-values very close to the cutoff (0.05) are considered to be marginal, and interpretation can go either way. Finally, p-values <= 0.05 indicates that the predicted motif has a strong significance
<table>
<thead>
<tr>
<th>Nucleotide Code</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Adenine</td>
</tr>
<tr>
<td>C</td>
<td>Cytosine</td>
</tr>
<tr>
<td>G</td>
<td>Guanine</td>
</tr>
<tr>
<td>T (or U)</td>
<td>Thymine (or Uracil)</td>
</tr>
<tr>
<td>R</td>
<td>A or G</td>
</tr>
<tr>
<td>Y</td>
<td>C or T</td>
</tr>
<tr>
<td>S</td>
<td>G or C</td>
</tr>
<tr>
<td>W</td>
<td>A or T</td>
</tr>
<tr>
<td>H</td>
<td>G or T</td>
</tr>
<tr>
<td>M</td>
<td>A or C</td>
</tr>
<tr>
<td>B</td>
<td>C or G or T</td>
</tr>
<tr>
<td>D</td>
<td>A or G or T</td>
</tr>
<tr>
<td>H</td>
<td>A or C or T</td>
</tr>
<tr>
<td>V</td>
<td>A or C or G</td>
</tr>
<tr>
<td>N</td>
<td>any base</td>
</tr>
<tr>
<td>. or -</td>
<td>gap</td>
</tr>
</tbody>
</table>

Table 2.1: IUPAC codes and corresponding value of bases

in the positive set.

**E-value** or expected value describes the number of times the predicted motif can be found by chance while searching in the dataset. It is a better metric of significance than p-value as it accounts for expectation. If the dataset contains only one sequence, then the E-value is the same as the p-value, but as the dataset gets larger, the E-value becomes larger. Hence, the lower the E-value, the more significant is the motif i.e. there is a lower probability of finding that motif by chance.

Circling back to the DREME [13] output, the p-values and E-values are used to select the significance of the motifs. The smaller the P-value, the more statistically significant is the motif and this is used to select the motif from a given set of predicted motifs.

**Position Weight Matrix (PWM), Position Frequency Matrix (PFM) and Position Specific Scoring Matrix (PSSM)** are some of the commonly used terms to represent motifs. This matrix attempts to capture the variable nature of the motif by indicating probabilities of finding a particular nucleotide at each position. This is done by first aligning sequences against each other followed by counting the number of times each nucleotide is found at each position. The matrix of counts is called as Position Frequency Matrix or PFM. From the PFM, the count
of each nucleotide is divided by the number of sequences aligned and the resulting matrix is called as Position Weight Matrix (PWM). Both these matrices assume positional independence as the probabilities is computed independently of other nucleotides in the motif. Position Specific Scoring Matrix (PSSM) and PWMs are often used interchangeably (See Figure 2.13 for an example of HuR PWM).

![Figure 2.13: Example PWM for HuR provided by Hilal et al.](image)

The highest scoring sequence is also known as the **consensus sequence**, and is sometimes reported as the binding motif by biologists; however alternative sequences can also be scored. PWMs provide a powerful tool for predicting the likely binding sites for any sequence.

Sometimes, RBPs recognize both primary sequence and secondary structure when binding to the target mRNA, and that makes the problem of identifying RBP targets different to that of identifying transcription factor binding sites.

### 2.5.2 Covariance Models (CMs)

Capturing secondary structure requires models to be built to identify patterns of covariation. Covariance models (CMs) are probabilistic models that have been used to model RNA families, multiple RNA sequence alignments, as well as consensus RNA structure prediction [18]. They are a special case of stochastic context-free grammar (SCFG). Context free grammars (CFGs) are a set of production rules that are used to generate patterns of strings. They have been used to describe RNA secondary structures, and Stochastic CFG (SCFGs) assign a probability of the production rules, and these probabilities are observed based on the training dataset. These models are built iteratively by aligning individual sequences to a single CM and refining the CM based on the alignment.

RNAPromo [147] is a recent covariance model-based computational tool that has been used
to identify both sequence and secondary structure. The algorithm takes as input a set of RNA sequences that are assumed to share a motif and any structural information about the motif if available. If no structural information is available, it uses the Vienna RNAFold [76] to fold the input RNA sequences, and the initial structure that is found is the most probable thermodynamically stable structure. RNAFold reads the RNA sequences and computes their minimum free energy (MFE) structure and this folding is carried out in sets of overlapping segments (200 base pairs in length with 100 base pair overlap). If the sequences are longer than 1000 bp, the amount of memory used increases with the length of the sequence. The memory requirements for folding a sequence of length \( n \) will be approximately \( 12 \times n^2 \). As longer sequences are folded, the amount of memory used increases and causes overflows. This tool generates a negative set of sequences (when not provided) by shuffling the positive set of sequences maintaining the dinucleotide distribution.

Cross-validation is used to learn the RNA motif. The input set is randomly partitioned into \( k \) sets (e.g. \( k = 5 \)), the tool learns an RNA motif from each of the \( k \) combinations of \( k - 1 \) sets. The tool then assigns a likelihood value to the set of RNAs that were held out when learning it and to the negative examples. Finally, there is one likelihood value for each real (positive) example and \( k \) likelihood values for each negative example. Assigning higher likelihood values to the held-out sequences compared with the negative examples suggests that there is a biological signal that is specific to the input RNAs.

RNAPromo [147] is a CM-based model for finding local motifs. It first identifies specific and relatively short candidate structures that appear in as many inputs as possible, and they are used to build an initial SCFG model. This model is then refined in the next step by using an expectation-maximization (EM) algorithm over the possible candidate structures. The algorithm iterates between a step of searching for the best alignment of the current SCFG model to the set of sequences and using the aligned positions to learn new parameters for the model. These candidate structures are then used as seeds for a probabilistic inference algorithm that refines the predicted motif using statistical estimation. The output of this computational method is a summary table displaying the graphical description of each motif, a file containing model parameters, and also a
file that summarizes the motif positions identified in each input sequence.

RNApromo\cite{147} has successfully predicted RNA motifs of three structural elements HFD, IRE, and SECIS. However, when further tested on predicting targets of RBPs the Puf family (Puf5, Puf4, Puf3) and HuR, RNApromo identified the target sites within the stem of a hairpin loop. Both these RBPs are known to bind to single-stranded regions of their targets \cite{121,207}. Further analysis carried out by Mukherjee et al. \cite{131} on HuR data suggested that the stem-loop binding model was inherently capturing biases in the di-nucleotide frequencies rather than secondary structure. The absence of secondary structure associated with HuR binding was further corroborated by Lebedeva et al.\cite{100}. They used Vienna RNA package 1.8.2 \cite{77} to computationally fold 3’UTR sequences within the 201 nt window centered around the 3’UTR binding sites. They used randomly chosen positions on those 3’UTRs as the control set. The base-pairing probabilities were averaged to find a substantially reduced pairing probability around the HuR anchors (peaks) but they did not find any indication of a hairpin structure.

Covariance models have some shortcomings; these methods require a large number of sequences (training examples) to build a good model. They are known to work well when the initial alignment is present to seed the search. However, these techniques fall short when considered as a technique to identify RBP targets. RBPs are known to have binding sites that range between 2-10 nts \cite{93}. The SCFG based methods, explicitly search for secondary structures and do not consider other regions around the structure that could affect the binding. The SCFG based methods will predict structure, but knowing when the structure is significant or relevant is not trivial.

This section reviewed some of the existing computational methods used to identify RBP targets. Motif-searching techniques fundamentally address primary sequence motifs and could potentially incorrectly predict RBP targets. Context-free grammar-based techniques during the selection of candidate motifs tend to assume the presence of secondary structure and ignore the regions flanking the structure. CMs tend to over-predict secondary structure. The interactions between RBP and mRNAs are not trivial and are non-linear in nature. SVMs, when used in association with kernel methods, are known to utilize the complex relationships that exist between
RBPs and associated mRNA targets. The next chapter will review the use of SVMs as an approach to identifying RBP targets.
Chapter 3

Computational Background

This chapter first introduces the identification of RBP targets as a classification problem. Support Vector Machines (SVMs) have been used to solve classification problems in the field of bioinformatics. Section 3.2 provides a brief overview of SVMs, followed by Section 3.2.1 that describes the \( k \)-spectrum kernel method as an approach to identify RBP targets. Section 3.3 reviews domain adaptation as a method to identify specific and shared binding preference of HuR and TTP.

3.1 RBP target identification as a classification problem

Predicting a given mRNA transcript as an RBP target can be expressed as a classification problem. Experimental methods discussed in Chapter 2 generate RNA sequences that are targets of the RBP that is being studied. Figure 3.1 outlines the computational pipeline involved to transform the sequencing data produced from experimental methods to generating sequences that can be used to build computational methods.

The reads obtained from the sequencer are typically short (around 20 nucleotides) in length. They are processed to remove any experiment method artifacts (such as 3’ adapter sequences added to the RNA fragments) to undergo further processing. These reads are aligned using an application called Bowtie [99]. Several of these short reads align to contiguous locations of the reference genome are grouped to create clusters. Clusters that contain only single reads are eliminated. These clusters are then annotated against a database of known transcripts and quality scores are computed based
**Figure 3.1:** Computational pipeline for PAR-CLIP analysis. Reads generated from experimental methods are processed to generate distinct reads. These reads and then aligned to the reference genome and quality filtered alignments are grouped into clusters. These clusters are scored based on various metrics and annotated against the reference genome. Appropriate cut-offs and quality scores are used to filter out false-positives and the resulting clusters are known to contain at least one binding site of the RBP being studied. This set is then subjected to statistical analysis and target peak calling. Reports are then generated in the forms of diagrams and HTML for each stage of the pipeline. Figure taken from [83]

on the number of unique reads alignments, the length of the cluster, and several other parameters. It is possible that the short reads obtained from the sequencer can get mutated by the effects of cross-linking and other experimental artifacts that take place during UV irradiation or RNA processing. A certain fraction of the reads result in false alignments and thus could lead to false positive binding sites. Other quality measures such as sense and anti-sense alignment of clusters are used to compute False Discovery rate (FDR) and clusters that score above the cut-off is filtered. Since the clusters are annotated against a known database of transcripts, the next step involves calling target genes associated with the clusters. The clusters could map into different regions of the RNA segment: coding region, 3´ UTR, 5´ UTR, introns, or exons. Coordinates of each cluster (start and end position), location the cluster mapped to, the strand, and other details are captured and stored in a BED file [145]. The process described in a nut-shell thus captures the mRNAs bound by RBPs in a cell at a given snapshot of time. It is straightforward to randomly select from the genome a set of negative sequences with similar length and/or composition that are known to not bind to the studied RBP. Using these two sets of sequences, it is possible to build a model that can discriminate the two classes, and furthermore be able to predict if a new set of sequences are binders or non-binders. Supervised learning algorithms in machine learning are good at solving
such problems.

In supervised learning, the data consists of training examples, and each example is a pair consisting of an input object (typically a vector), and an output value or a label associated with the input object. Classification problems typically contain two or more classes; for example, a positive class consisting of sequences that contain RBP targets generated from experimental methods discussed in Chapter 2, and negative class that is sequences that are known to not bind to the RBP. The supervised learning algorithm analyzes the training data (both positive and negative training examples) and creates a model (classifier) which is then used to label unseen data (test data). The goal of the classification model is to minimize the number of classification errors on future unseen or test data. There are different machine learning approaches to build the model. However, we used Support Vector Machines (SVMs) as they have been successfully used in solving problems in the field of bioinformatics. The next section will provide a brief overview of SVMs.

3.2 Support Vector Machines and Kernel Methods

SVMs learn from training examples where each example is a pair consisting of an input object also called as the “feature vector” and an output value - a label associated with the input object. The machine learning algorithm analyzes the training data (both positive and negative examples) and creates a model (classifier) which is then used to classify unseen data. SVMs construct a hyperplane in a high or infinite dimensional space and the hyperplane is used for classifying the positive and negative data [39].

More specifically, given data: \(((x_1,y_1),\ldots,(x_N,y_N))\), finding the separating hyperplane is posed as a constraint satisfaction problem as shown in Figure 3.2. For all \(i\) from 1,.., \(N\), find \(w\), such that \(w.x_i + b \geq +1\) if \(y_i = +1\) and \(w.x_i + b \leq -1\) if \(y_i = -1\) [39]. The vector \(w\) is called as the weight vector and the scalar \(b\) is known as the bias. If the data is linearly separable, there are an infinite number of hyperplanes that would satisfy the constraint. However, the best classifier would be the one that would maximize the perpendicular distance (margin) between the nearest training data points of any class. The training data points that are on the hyperplane are called
support vectors (See Figure 3.5). The optimal hyperplane will be the one with the largest margin. Intuitively, a model with few support vectors without compromising the quality of the classifier is beneficial — leaving out a training example that is not a support vector would give the same solution. A large amount of support vectors could be indicative of over-fitting especially if the number of support vectors are close to the number of examples.

**Figure 3.5:** A: Large Margin SVM: applicable to linearly separable data. $w$ and $b$ are computed by solving the quadratic equation $\min \frac{1}{2} \|w\|^2$ s.t $y_i (w \cdot x_i + b) \geq 1, \forall x_i$ and minimizing $\|w\|^2$ is equivalent to maximizing the margin, B: Soft-Margin SVM: applicable when data is not linearly separable. Slack variable $\zeta$ is added that allow an example to be misclassified. The optimization equation thus becomes $\min \frac{1}{2} \|w\|^2 + C \sum_i \zeta_i$ subject to $y_i (w \cdot x_i + b) \geq 1 - \zeta_i, \forall \zeta_i \geq 0$. The constant $C > 0$ dictates the relative importance of maximizing the margin and minimizing the amount of slack, C: Kernel method mapping. Figures is taken from [157]

When the sets are not perfectly separable, SVM finds a hyperplane that maximizes the margin and minimizes the misclassification (See Figure 3.3). This concept is also known as “soft”-margin SVM where some examples can be misclassified but at a cost which is called as slack variable $\zeta$. The SVM parameter $C$ plays an important role in determining the width of the margin. A lower value of $C$ results in a larger margin of separation and could lead to points that are misclassified whereas a higher value of $C$, results in a smaller margin, but most of the points will end up being classified correctly. Selecting this value of $C$ depends on the requirement of how important it is to classify every training example accurately.

However, there are cases when non-linear regions can achieve more efficient separation. In such cases, SVM handles this by using a kernel function that maps the data into a higher dimensional space where the sets are better able to be linearly separated. SVMs use a mapping into a larger space so that the dot product can be computed easily in terms of variables in the origi-
nal space, thus making the computational load reasonable. Kernel functions represent similarity between two objects and are defined by the dot product in the vector space.

SVMs are regarded as a “black-box” technique and interpreting results are not always intuitive. The mechanisms used by this technique to transform the input into output is hidden from the user. While understanding the mathematics underlying the decision making can be hard, it is dangerous to apply “black-box” methods blindly. However, SVMs have been used successfully to solve several problems in the field of bioinformatics [159]. While this could be attributed to the performance of SVMs, but perhaps also due to the fact that the SVM algorithms have been implemented in several popular programming languages such as R and Python. This has led SVMs to be adopted by a much wider audience who might have passed it possibly due to the complex math involved with the implementation.

Black-box techniques provide an abstraction of the underlying model (model agnostic) which results in flexibility on the type of machine-learning technique used to solve the problem at hand. In such systems, information hiding is constrained by the complexity of the implementation. The implementation can choose the right method based on the input received from the user. Such models can handle data complexity such as image, audio, and sensory data processing. Also, the underlying model can be switched from under the user but the result is guaranteed a certain performance. Black-box techniques can be used to solve problems in systems that require a certain level of performance to be guaranteed but can forgo the understanding the reasons behind the decision-making.

On the flip side, model abstraction or inability to understand why certain decisions are made can cause users’ mistrust of systems, users’ inability to contribute to the maintenance of correct and consistent data, and finally unable to understand the inner working of the system. Building a transparent system could possibly incur higher cost, where every decision would have to be explained to the user. Bayesian networks, rule-based systems, random forests, decision trees are some examples of transparent systems as they provide insights into the decision being made by the model. Random Forests are an ensemble learning method that operates by constructing multiple
decision trees during training, and the output is the predicted class that is the mode of the classes 
in the case of classification) and mean prediction (in the case of regression) of the individual trees. 
The algorithmic goal of random forests is to create many trees and inject randomness into each 
one so that they each have the maximal strength and minimal correlation. The algorithm used to 
build $K$ trees:

(1) Build each tree from a bootstrap sample, which is $N$ data points chosen randomly with 
replacement from $N$ data points. Each tree is built by:

- Selecting, randomly, at each node a small set of features, $F$, to split on. $F$ is kept 
constant through the $K$ trees.
- Then for each node, we split on the best of the features in set $F$.
- Let the tree grow to full length.

A random set of features are selected at each split possible in the learning process with the 
intuition, if one or a few features are strong predictors for the target output, the features would 
be selected in many of the $K$ trees. By permuting through the features to build the classifier, the 
impact of the features on the accuracy of the classifier can be studied. Thus providing insights 
into the features first used for classification and then their impact on the resulting accuracy of 
the classifier. Additionally, feature selection can be incorporated to improve the accuracy of the 
classifier built. Similarly, in the case of decision trees, for a given input, traversing the trees helps in 
understanding the final outcome of the classifier. When compared to the SVMs, where the classifier 
is selected in a truly “black-box” manner.

SVMs are also prone to over-fitting the data but this can be mitigated by careful tuning of 
the regularizing parameter $C$. Smaller the value of $C$, the risk of over-fitting can be reduced as 
the value of $C$ controls the width of the margin. The smaller the value of $C$, the wider is the 
margin, the more and larger in sample classification errors are permitted. SVMs can also incur 
high computation costs as the maximum margin hyperplane depends on the square of the number
of training examples. It also requires a larger amount of storage, as all the support vectors need to be stored.

However, being able to define similarity features and map the sequences in the vector space makes kernel methods attractive. Using kernel methods in conjunction with SVMs, it is possible to handle non-vector inputs such as variable length sequences, protein structures, or gene networks—a common source of input data in computational biology applications. Another advantage of using kernel functions is computational cost since the size of the kernel matrix depends only on the number of training examples and is independent of the dimension $d$ the data is mapped into. Kernel methods have evolved in the field of computational biology and as a result have been able to keep up with high-dimensional data. String kernels encode the sequences into a vector representation, as well as are able to identify features which translate to binding sites and/or motifs. An example of such string kernel is the $k$-spectrum kernel which has been proposed to classify protein sequences [102, 103] and will be reviewed in the next section.

### 3.2.1 $k$-spectrum kernel method

The basic idea of a string kernel [103] is to count the number of subsequences of length $n$ in a string, and compose a high-dimensional feature vector by these counts. In this kernel, the feature space consists of short sub-sequences of length $k$ also known as $k$-mers. Thus the $k$-spectrum of a biological sequence is a set of all $k$-length sub-sequences that it contains. A feature map is defined as the number of times a particular $k$-mer appears in the string and is indexed by all possible $k$-length subsequences. In the case of protein sequences, the dimension of the feature space is $20^k$. This kernel method defines sequence similarity by considering the number of $k$-mers that two sequences share. The larger the number, the higher the $k$-spectrum kernel value.

Figure 3.6(a) [102] is an example of the $k$-spectrum for the string **AKQDYYYEI**. The feature map for this $k$-spectrum is displayed in Figure 3.6(b) [102], and more formally defined as [102]:

$$\Phi_k(x) = \theta_\alpha(x) \alpha \epsilon A^k$$

(3.1)

where, $\theta_\alpha(x)$ is the number of occurrences of $k$-mer $\alpha$ in the sequence $x$. The $k$-spectrum kernel
of two sequences $x$ and $y$ is obtained by taking the dot product in the feature space

$$K_k(x, y) = (\Phi_k(x), \Phi_k(y))$$

Equation 3.2 does not consider any mismatches in the $k$-mers, a case that is seen commonly in biological sequences. Leslie et al. [102] proposed an extension to this kernel method called as the mismatch kernel which is biologically more realistic. Thus two $k$-mers are considered similar if they have at most $m$ mismatches (See Figure 3.6(c)). More formally, for a fixed $k$-mer $\alpha = a_1, a_2, ..., a_k$, where each $a_i$ is a character in $A$, the $(k,m)$-pattern generated by $\alpha$ is the set of all $k$-length sequences $\beta$ belonging to $A$ that differ by at most $m$ mismatches. This set of sequences is also called the mismatch neighborhood. The feature map for the sequence $x$ will be the sum of all $k$-mers that satisfy the $(k,m)$ condition.

For example, the feature map for the $k$-mer AKQ (Figure 3.6(c)), is the set of all sequences of length 3 that differ from AKQ by at most 1 mismatch. This groups the $k$-mers that are similar together.

If $\alpha$ is a fixed $k$-mer, the kernel function $\Phi_{(k,m)}$ on $\alpha$ is defined by [102]:

$$\Phi_{(k,m)}(\alpha) = \sum_{k\text{-mers } \alpha \text{ in } x} \Phi_{(k,m)}(\alpha)$$

When $m$ is zero, $\Phi_{(k,0)}$ corresponds to $k$-spectrum kernel defined earlier in equation 3.2. The $(k,m)$
mismatch kernel for sequences \(x\), and \(y\) is the inner product in the feature space:

\[
\Phi_{(k,m)}(x,y) = (\Phi_{(k,m)}(x),\Phi_{(k,m)}(y))
\] (3.4)

The kernel value \(K_{(k,m)}(x,y)\) will be large if the sequences \(x\) and \(y\) share many \(k\)-length subsequences differing by at most \(m\) mismatches. The data structure used to compute the kernel value is described below.

**Trie-data structure:** The \(k\)-spectrum kernel method computes the kernel matrix using a mismatch tree data structure (similar to trie or suffix tree). This structure is used to represent the set of all \(k\)-mers (feature space) and it is traversed to obtain all instances of \(k\)-mers that occur within \(m\) mismatches in the data set. The kernel matrix is computed using one tree traversal. The \((k,m)\)-mismatch is a rooted tree of depth \(k\) (length of \(k\)-mer with \(l\) branches). The leaf nodes represent a fixed \(k\)-mer in the feature space. This \(k\)-mer is obtained by concatenating the symbols along the path from the root to the leaf. A depth-first search is performed and at a node \(d\), \(n\) pointers are stored to all the \(k\)-mer instances from the data set whose \(d\)-length prefixes are within \(m\) mismatches from the \(d\) length prefix represented by the path down from the root. Once a node with an empty list of pointers is encountered, there is no longer a need to search below that node in the tree indicating that there are no valid occurrences of the current prefix. The kernel values are updated at each leaf node. This value is computed by adding the contribution of all instances of the \(k\)-mer occurring in each sequence to get the feature values corresponding to the \(k\)-mer. As seen in Figure 3.7(a) and 3.7(b), the kernel value \(K(x,y)\) is computed by adding the contribution of the \(k\)-mer ADL.

The efficiency of this kernel method is in the way the kernel value are computed. There is no need to store the mismatch tree and this makes efficient use of memory and makes computing large data sets possible. The number of \(k\)-mers with \(m\) mismatches of a given fixed \(k\)-mer is

\[
p(k,m,l) = \sum_{i=0}^{m} \binom{k}{i} (l-1)^i = O(k^m l^m).
\]

The number of \(k\)-mers that need to be traversed grows with the complexity of \(O(Nk^m l^m)\) where \(N\) is the total length of the sample data. The number of kernel matrix updates done if \(c\) input sequences contained a \(k\)-mer is \(c^2\) and the worst case running time for \(M\) sequences each of length \(n\) occurs when \(M\) feature vectors are all equal and
have maximal number of non-zero entities is $O (M^2 np(k,m,l)) = O (M^2 nk^m l^m)$. Thus, this number is small if $m$ is small.

When the mismatch kernel is used in combination with SVMs, the learned classifier $f(x) = \sum_{i=1}^{r} y_i \alpha_i \langle \phi_{(k,m)}(x_i), \phi_{(k,m)}(x) \rangle + b$, where $x_i$ are the training sequences that map to the support vectors, $y_i$ are the labels, and $\alpha_i$ are weights can be implemented by pre-computing and storing per $k$-mer scores.

While this section focused on the $k$-spectrum kernel method, other kernel methods used in the field of computational biology are reviewed extensively by Schölkopf et. al [159]. Highlights of that review are discussed in the subsequent section.

### 3.2.2 Applications of Kernel Methods in Computational Biology

This section is focused on string-based kernel methods that have been successfully used in the field of bioinformatics. The first successful use of SVM-based approach in the field of protein homology detection was seen in the late 90s. Jaakkola et al. [81, 82] improved on the accuracy of classifying protein sequences by modeling the difference between protein sequences (positive examples) and unrelated sequences (negative set). The proposed method was called SVM-Fisher [82] method which trains a Hidden Markov Model (HMM) on data to obtain a Fisher kernel for an SVM. This feature-based kernel method improved accuracy by modeling the differences between
positive and negative examples. This implementation was compared to Basic Local Alignment Search Tool (BLAST) \[128\] (an initial method used to classify protein families) and generative HMMs \[85\] and the experiments measured the success rate in identifying the member of superfamilies of the SCOP \[136\] protein structure classification. The SVM-based method improved on classifying protein families by three times when compared to two implementations of BLAST and identified 10 more families when compared to the HMM approach. Variable length sequences pose a challenge as there isn’t a way by which a sequence can be converted to a vector. However, this can be handled by extracting numerical features from strings and then creating a kernel by computing a dot product between the resulting feature vectors. For example, the spectrum kernel has been proposed to classify protein sequences \[102, 103\] and is discussed in the previous section. When the mismatch kernel method was run on the same SCOP dataset \[136\], the \(8, 2\) mismatch kernel out-performed BLAST and Smith-Waterman techniques in any remote homology/fold detection tasks \[103\]. Beer et al. \[101\] demonstrated that the use of a \(k\)-mer based (using DNA sequence elements as features) approach can be used to distinguish enhancers from random genomic regions. Schultheiss et al. \[160\] successfully developed a Python pipeline called KIRMES using the \(k\)-spectrum kernel to identify degenerate motifs from microarray data in Arabidopsis thaliana. The advantage of using the \(k\)-spectrum kernel method for classifying sequences is the feature \(k\)-mers identified by this method could be potential candidate binding motifs.

SVMs have been successfully used to predict protein-protein interactions as well as predicting protein-centric interactions with target mRNAs \[35, 73, 166\]. Han et al. \[73\] have used an SVM to discriminate between a set of RNA-binding proteins and non-RNA-binding proteins based on their primary sequence. Furthermore, the SVM model training in this work has published as a web-base protein functional classification software SVMProt \[29\]. Kumar et al. \[95\] have combined evolutionary information to build an SVM model to predict RNA binding sites in a protein sequence. The evolutionary information is captured in the form of Position Specific Scoring Matrix (PSSM) profiles of the protein sequences were provided as input to the SVM model, and this model has been trained and tested on 86 RNA binding proteins and evaluated using five-fold cross validation.
However, SVMs have not been explored as an option to predict targets of RBPs. Given the success of SVMs and the $k$-spectrum kernel discussed above in applications that are similar to RBP target identification, prompted me to explore this as an approach to identify RBP targets.

Most RBPs contain multiple individual domains and hence could share binding preferences. Competitive binding is also known to exist between multiple RBPs. There exists evidence for competitive binding between HuR and TTP [180] as well as these RBPs are also known to recognize similar binding sites (AREs) [100, 130]. Multi-task learning (MTL) has been used to build classifiers that can transfer knowledge between domains and are able to identify specific and shared features of each domain. The next section reviews MTL and domain adaptation concepts.

### 3.3 Multi-task learning (MTL) and Domain Adaptation

Multi-task learning (MTL) is a machine learning method that learns a problem from multiple related tasks at the same time, using a shared feature representation. The goal of MTL is to improve the performance of learning algorithms by learning classifiers for multiple tasks jointly and works well if these tasks have some commonality [193]. There exists three high-level strategies for MTL: (i) instance-based transfer, where data points from different domains are included into the learning problem, typically in combination with some form of re-weighting, (ii) feature representation transfer, where the instances from the various domains are mapped into joint feature representation, (iii) parameter transfer, parametric learning paradigm which is based on the assumption that closely related tasks should also yield similar parameters in the learning model. Since HuR and TTP share binding sites, as well as exhibit similar binding preferences, feature augmentation technique is explored to create a combined model to identify shared features as well as specific features of each RBP.

The goal of domain adaptation is to develop algorithms which are able to transfer from one domain to another. In domain adaptation, each task is associated with its own domain, and the goal is to improve the performance of the classifier on the target domain. To build models, three different scenarios require being considered: (i) out-of-domain (source only) where the model is
trained on source data only and test the model on target data. (ii) in-domain (target only) where
the model is trained on target data only, and (iii) data combination (union of a source and target)
which involves training the data on both source and target domains. An example of out-of-domain
baseline, the model is built using HuR sequences (source-domain) but tested on examples from the
TTP domain, i.e. given a set of TTP binders and non-binders, how accurately can the HuR model
predict them? This is considered as a baseline model before applying any adaptation techniques.

Domain adaptation techniques are very popular in natural language processing (NLP) ap-
plications. In NLP, the “source domain” could comprise of labeled data in one domain (such as
news articles) and the goal is the classifier built using the source data should be able to perform
well when classifying in a target domain which could be completely different to the source domain
(e.g. classify emails as spam or valid content). There are two different types of domain adaptation
techniques: Supervised domain adaptation and Semi-supervised domain adaptation. In supervised
domain adaptation, there is labeled data available for both source and target domains and the
goal is to build a model leveraging both the domains that perform well in the target domain. In
semi-supervised domain adaptation, labeled and unlabeled data is available in the target domain.
The focus of this thesis is the supervised domain adaptation using the feature space augmentation
technique discussed in the subsequent section.

3.3.1 Domain Adaptation: Feature Space Alteration

To identify features specific and shared by domains, Daume III [40] proposed feature aug-
mensionation approach wherein the feature space is tripled (when the number of domains=2). For
each feature in the original feature space, there is a “general” version or “original” version, a
“source-specific” version, and a “target-specific” version. The underlying learning algorithm is
then expected to “learn” which features transfer across domains (hence shared features) and which
do not (specific features).

Formal explanation of this concept it taken from Daume III [40] and presented here : $\chi$ and
$\upsilon$ are the input and output spaces respectively, and $D^s$ is the source domain data-set and $D^t$ is the
target domain data-set. Suppose, $\chi = \mathbb{R}^F$ for $F > 0$ and $v = -1, +1$. The augmented feature space thus is $\tilde{\chi} = \mathbb{R}^{3F}$. The mappings for source and target data $\Phi^s, \Phi^t$ can be defined as: $\chi \rightarrow \tilde{\chi}$. Thus,

$$\Phi^s(x) = \langle x, x, 0 \rangle, \Phi^t(x) = \langle x, 0, x \rangle$$ (3.5)

The feature $x_i$ in the source domain is duplicated into three versions: $\langle x_i, x_i, 0 \rangle$ where the first one is source domain specific, the second one refers to shared or domain independent version, and the last one is target domain specific version. Since $x_i$ was from the source domain, the last column in the example is 0. Similarly if $x_i$ was a target domain specific feature, the corresponding feature vector would be $\langle 0, x_i, x_i \rangle$. The new feature vectors are fed into a supervised learning algorithm, and the algorithm learns which feature vectors transfer across domains and those that do not. The augmented feature space for $K$ domains will contain $K + 1$ times the number of copies of the original feature space.

Ease of implementation is one of the advantages of this technique. Also, this method can be applied to several domains. Each feature is duplicated for each domain, plus a general version of the feature is added. This can be used with any statistical classifiers or learning algorithms as it does not rely on any knowledge from the classifier. However, this approach does have limitations. It is dependent on training data in every domain, and the performance is dependent on the size of the target domain training data. Also, there is no way to handle specific nature of domain-specific features if they exist. Additionally, based on the number of domains, the feature space could grow linearly. Finally, this approach does not make any distinction or connection between domain-specific and domain-independent features. This limitation is overcome by applying prior knowledge in choosing features and is explained in the next section.

3.4 Domain Adaptation: Apply Prior Knowledge

Following the work of Daume III [40], Finkel et al. [55] proposed a slight modification to the “preprocessing step” where the hyper-parameters were separated and were set to different values from each other. Similar to the model proposed by Daume III [40], each feature weight ($\lambda_i$) is replicated; “domain-specific” version and “general” version. However, there is a slight difference;
rather than a constant prior over these feature weights, there are two different types of gaussian priors: domain-specific prior and the second is domain-independent prior. The prior of each domain \( d \) controls the feature weight for this domain \( \lambda_{d,i} \), and the prior mean is the domain-independent version of the feature weight \( \lambda_{s,i} \). The \( \lambda_{s,i} \) is controlled by the top-level domain-independent prior with zero mean. Unless there is domain-specific evidence (feature only exists in a particular domain, and is guaranteed to not exist in the other domains), the prior makes features have similar weights across domains. The inherent hierarchy that exists in the model allows the feature weights in different domains to leverage each other through the top-level domain-independent prior. The domain-independent parameters \( \lambda_{s,i} \) are used as the prior mean of the domain-specific parameters, they can influence the value of domain-specific parameters \( \lambda_{d,i} \). If there is a strong evidence of a feature for a specific domain \( d \), the model forces \( \lambda_{d,i} \) to be a large value, and if there is little knowledge of the feature being domain-specific, the \( \lambda_{d,i} \) value will be set to a lower value. It is important to note that Finkel et. al demonstrate that the model proposed by Daume III \[40\] is a special case of this model when the domain-specific prior mean is 0 and the variance \( \sigma_d = \sigma_s \). The main advantage of this approach is its ability to support multiple domains unlike that proposed by Duame III \[40\].

Going back to our example in the previous Section 3.3.1, if the feature \( k \)-mer “AUUAUUA” is shared by both the domains, the feature weight associated with that feature will be the same in both HuR and TTP domains. However, the feature “UUUUUUU” will be assigned a high (positive) value in the HuR domain (since it is HuR-specific), and low (negative) value in the TTP domain. Thus, the domain-specific evidence will out-weight the effect of the prior.

So far, this chapter provided the computational overview of the methods that are used in this thesis to identify RBP targets. Since this thesis is also focused on sequence generation and benchmarking, the next section will go over generative models as an approach to generate synthetic datasets.
3.5 Generative models for sequence generation

SVMs are an example of a discriminative model (Section 3.2) which take as input training
eamples $x_i$ where $i = 1,2,...n$ and the corresponding labels $y_i$, and are able to output the most
likely value of $y$ for a future unseen $x$. Such models estimate the probability $P(y|x)$ directly.
Discriminative models are able to classify unseen future data but do not have an understanding of
how the input data is generated.

Generative models are able to generate observations for a particular domain of interest based
on probabilistic rules that govern the domain along with some hidden variables. They are able
to learn a function $f(x,y)$ for an underlying dataset and create a scoring scheme between $x$ and
$y$. In the bioinformatics domain, generative models are used to generate synthetic datasets that
have characteristics of a particular region of the genome (3’ UTRs, coding regions, introns, etc.).
These models are not only able to generate datasets, but also can classify unseen data. They learn
a function $f$ that takes a sequence that maps inputs $x$ to labels $f(x)$ and do so by learning a
joint distribution $P(x,y)$ over inputs $x$ to labels $y$. Using Bayes rule, this expands to: $P(x,y) =
P(y)P(x|y)$ where $P(y)$ is considered as a “prior” (measure of prior likelihood) and $P(x|y)$ is the
conditional generative model which gives the probability of $x$ given $y$. Generative models learn the
joint distribution $P(x,y)$ and when are used to classify a new training example $x$.

For a given input example $x$, the $f(x)$ is defined as $y$ that maximizes the conditional probabil-
ity of $P(y|x)$ (refer to Equation 3.6). An important characteristic of Equation 3.6 is the denominator
$P(x)$ does not vary with $y$ and hence that term can be discarded when evaluating $\arg\max_y$.

\[
f(x) = \arg\max_y P(y|x) \\
= \arg\max_y \frac{P(y)P(x|y)}{P(x)} \\
= \arg\max_y \frac{P(y)}{P(x|y)}
\] (3.6)

Some of the examples of generative models are Naive Bayes, Hidden Markov models (HMMs),
Latent Dirichlet Allocation (LDA). The next section provides a brief overview of HMMs and their
use in the field of computational biology.

### 3.5.1 Hidden Markov Models

HMMs are a statistical modeling method and is composed of a number of states. Each state emits symbols that are governed by emission probabilities. These states are inter-connected by state transition probabilities. The start or initial state is entered, followed by a sequence of states based off state transition probabilities, and finally, the end state is reached. At each state, symbols are emitted (observed event) and the state sequence corresponds to the hidden parameters. The sequence of states is called as a Markov chain but since the state sequence is hidden and only the symbols that are emitted, it is known as Hidden Markov Model.

HMMs have been used extensively in speech recognition [36, 148] where the symbols emitted are the sounds that form the word, and the hidden model generates the highest probability sound. When used to generate biological sequences, a nucleotide at a particular position depends only on the state in which the previous nucleotide position is in the sequence. To model sequences in bioinformatics, an HMM can be constructed to model regions (that correspond to states) such as AT-rich, GC-rich, intron, exon, promoter, and the probability of emitting nucleotides “A”, “T”, “G”, “C” differs based on the state that the HMM is in. Figure 3.8 illustrates an example HMM that models AT-rich and GC-regions of sequences. The state transition probabilities describe the likelihood of switching states, for example in Figure 3.8 we see that the transition probabilities of switching from AT-rich state to GC-rich state is 0.3, and vice-versa is 0.1. When in the AT-rich state, the probability of emitting nucleotides A and T are much higher than that of emitting G and C. Similarly, in the GC-rich state, nucleotides G and C will be emitted at a higher probability. Switching between AT-rich and GC-rich states is governed by the transition probabilities. The state transition probabilities are stored in a matrix called HMM transition matrix.

Building an HMM from a set of training sequences is trivial if the emission and state transition probabilities can be estimated by calculating the expected value by evaluating the number of times state is changed divided by the sum of all transition probabilities, and a particular
Figure 3.8: Example of a HMM model used to generate AT-rich and GC-rich sequences [37]

symbol is emitted divided by the sum of all the emission probabilities. However, when the state paths are unknown, the model parameters need to be estimated iteratively in order to maximize the probability of all sequences in the training set. An iterative Expectation-Maximization (EM) algorithm, known as the Baum-Welch [17] algorithm is used to estimate model parameters.

HMMs are also used to estimate the optimal sequence of hidden states as well as computing the likelihood of a given data for a particular set of model parameters and observed data. Both these problems can be solved using dynamic programming algorithm known as the Viterbi algorithm [188] and Forward-Backward algorithm. Krogh et al [91] introduced a novel method of modeling protein families using HMM. In this paper, the authors describe the HMM architecture used to estimate parameters from a set of training sequences, use of HMM to obtain multiple alignments (using the Viterbi algorithm), extending HMMs by iteratively splitting the sequences into clusters of similar sequences and developing models for each protein subfamilies, and finally using HMMs to model protein domains.
Yoon et al [203] review the use of HMMs in the field of bioinformatics and their use to discern between different regions of the DNA (coding vs. non-coding regions), protein homology detection [169], model protein binding sites [94], multiple sequence alignment [47, 15], searching for protein families in databases, and an attempt to model RNA secondary structure [74]. Analyzing the long-range pairings that exist in secondary structure portions of RNA sequence is challenging when using HMMs. However, SCFGs (described in Chapter 2, Section 2.5.2) are able to describe long-range interactions and are typically used when modelling RNA secondary structures [44, 47].

This section provided a brief overview of generative models and their use in the field of bioinformatics. Chapter 5 discusses the model used in this thesis to generate sequences. The use of the \( k \)-spectrum kernel method to identify RBP targets as well as domain adaptation technique to identify shared and specific binding targets of RBPs is explored in the next chapter.
Chapter 4

Discriminating between HuR and TTP binding sites using the k-spectrum kernel method

4.1 Motivation

As seen from Chapter 2, both HuR and TTP bind to similar AU-rich elements (AREs) which are generally 50–150 nucleotides long and usually located within the 3′ untranslated region (UTR). The targets they recognize are very different in lengths; HuR is known to recognize longer 3′ sequences, whereas TTP much shorter sequences (See Appendix G, Figure G.1). The stability of the mRNA is known to be dependent on the length of the 3′ UTRs. Genes with more microRNA (miRNA) sites are known to have longer 3′ UTR lengths [172] and harbor more binding targets per KB of the 3′ UTR sequence. In addition, by binding to specific sites with the 3′ UTR, miRNAs could decrease gene expression of various mRNAs by either inhibiting translation or directly causing degradation of the transcript. When over-expressed in cells, HuR stabilizes ARE-containing transcripts and promotes their translation. One such example is HuR’s role in the formation of new skeletal muscle tissue. HuR is known to target two key muscle regulatory factors (MRFs), MyoD, and myogenin, and the binding leads to mRNA stability and muscle cell differentiation [185]. TTP is a nucleo-cytoplasmic shuttling protein that interacts with AREs through tandem CCCH zinc fingers (TZFs) [97]. The interaction of TTP with AREs in the 3′ UTR of targeted mRNAs, such as TNFα, promotes mRNA degradation [79].

Previous efforts on understanding the AU-rich binding proteins HuR and TTP were focused primarily on identifying k-mer motifs. Mukherjee et al. [130] demonstrated that over 80% of the
TTP sites in 3′UTRs overlap with HuR target binding sites. The motif recognized by HuR is more U-rich whereas the motif recognized by TTP is predominantly AU-rich [130]. However, it remains unclear to what extent these features discriminate between HuR and TTP binding in vivo.

Position-specific scoring matrix (PSSM)-based approaches are commonly utilized to describe protein binding [22, 64, 127, 192]. These methods assume positional independence within the motif, which is sometimes also represented as a position frequency matrix (PFM). Another representation of motifs is $k$-mer based or consensus strings. This representation can capture intra-motif dependencies that are missed with the PSSM-based approach. Machine learning approaches that utilize $k$-mer searches factor in these dependencies [21, 78, 110, 164], and in doing so can provide more discriminative power.

The $k$-spectrum kernel method has been used successfully in a number of bioinformatics applications. In protein sequence classification [103] it out-performed both BLAST [6] and Smith-Waterman [168] at super-family homology and fold recognition tasks. Beer et al. [101] demonstrated that the use of $k$-mer based (using DNA sequence elements as features) approach can be used to distinguish enhancers from random genomic regions as well as short transcription factor-binding sites for tissue specificity. Schultheiss et al. [160] successfully developed a Python pipeline called KIRMES using the spectrum kernel to identify degenerate motifs from micro-array data in Arabidopsis thaliana. Hence, in this chapter, I explore the use of the $k$-spectrum kernel method to first identify RBP targets of HuR and TTP, followed by using domain adaptation technique to identify specific as well as shared features of these RBPs.

4.2 PAR-CLIP HuR and TTP clusters

Two papers have explored the binding targets of HuR and TTP. Lebedeva et al. [100] used PAR-CLIP [72] on unstressed HeLa cells to identify the binding targets of HuR. Reads were aligned and clusters of continuous read coverage were identified. Clusters were then filtered based on detectable nucleotide conversion events (a side effect of the crosslinking) and a quality score metric. The authors classified hits as consensus if detected in two out of three PAR-CLIP [72] experiments.
and **conservative** if detected in all three experiments. We utilized the conservative data-set for computational analysis such as identifying motifs. We obtained the data set as a BED file mapped to the hg19 genome from doRiNA [8]. Mukherjee et al. [130] carried out PAR-CLIP [72] experiments to identify the genome-wide binding sites for TTP. Reads were mapped to the genome and binding sites identified [38]. It is worth noting that Mukherjee directly compared their binding sites for TTP to previously published HuR data [129] using a consistent analysis pipeline. The final target list includes 4,626 peaks of mRNA-TTP interactions, downloaded from GEO (Gene Expression Omnibus) accession number GSE53185. For both data sets (positive set), peak statistics (minimum, average, and maximum length) are provided in Table 4.1.

<table>
<thead>
<tr>
<th>Number of Sequences</th>
<th>Max length</th>
<th>Min Length</th>
<th>Average Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>HuR</td>
<td>3642</td>
<td>243</td>
<td>19</td>
</tr>
<tr>
<td>TTP</td>
<td>4626</td>
<td>172</td>
<td>21</td>
</tr>
</tbody>
</table>

### 4.2.1 HuR and TTP experimental methods

**HuR dataset:**

- Unstressed HeLA cell were used in three PAR-CLIP experiments
- 4SU in normal and 6SG in SILAC and normal medium
- Reads were aligned to the hg19 genome co-ordinates
- SAMTools [104] was used to generate pile-up file
- Custom scripts were run on the pileup output to identify clusters (continuous read coverage)
- Clusters were scored based on: a) Number of conversion events (T-C and G-A) and b) Number of overlapping reads with variable start and end positions in the same cluster were assigned higher quality score
• Within the clusters, the position with higher number of conversion events was computed which coincides with the preferred position of cross-linking.

• 7-mers with less than ten occurrences were discarded and they compared the frequency of the 7-mers with the RefSeq [143] 3UTR sequences as background set.

**TTP dataset:**

• Reads were aligned to the hg19 genome co-ordinates.

• SAMTools [104] was used to generate pile-up file.

• Reads that were less than 20 nt were discarded.

• Used T-C and G-A number of these conversion events as quality measure.

• Reads with T-to-C mismatches were gathered using PARalyzer [38] to retain only those that mapped to a single genomic location with the number of mis-matches as the criteria.

• Computed the signal to noise ratio (SNR) by calculating the occurrence of known RNA recognition elements (RRE) in the TTP mRNA sites relative to the background set (3UTR sequences). The RREs were 40-250 times more enriched than the background set.

• The AUUUA pentamer was relatively more enriched in the TTP sites whereas the UUUUU and U-stretches flanked by C on either side were enriched in the HuR sites.

4.2.2 Control sequences

For both the HuR and the TTP data-sets, a negative set of control sequences were generated using random 3’UTR transcripts from the human genome. For each binding sequence in the positive set, a size matched negative sequence was selected at a random location from a random transcript (not in the positive set). One control set was generated for each RNA binding protein.
4.2.3 Computation tools for motif validation

To verify the motifs identified by Lebedeva et al. [100] and Mukherjee et al. [130], two primary sequence motif discovery tools were utilized. Discriminative regular expression motif elicitation (DREME) [13] is part of the MEME (multiple EM for motif elicitation) [12] tool suite (v 4.10.2) that uses a discriminative approach to motif discovery. DREME [13] was utilized with a motif length range of 7-13 nucleotides. DREME outputs the discriminatory \( k \)-mers along with E-value, p-value, and the position frequency matrix (PFM). The top five \( k \)-mers based on E-value were short-listed for comparison to the string kernel results [13]. The results list the over-represented \( k \)-mers (based on a number of occurrences) associated with identified motifs. A second approach is the use of a string kernel function [102], a method that groups sequences with similar \( k \)-mers. This method has been used effectively to determine homology of protein sequences that share a remote evolutionary relationship [178]. The \( k \)-spectrum kernel method was implemented using the PyML library.

For the \( k \)-spectrum kernel, the HuR, and TTP target sequences were each used independently to build \( k \)-spectrum kernel predictive models using an SVM. For each RNA binding protein (HuR and TTP), the model was built using the 80% of the sequences by iterating over three parameters; the SVM parameter \( C \) (cost of a mislabeling) and \( k \)-spectrum parameters \( K_1 \) and \( K_2 \) (\( k \)-mer length). The remaining 20% was held out as a validation set. The parameters \( K_1 \) and \( K_2 \) ranged between values 7 and 13 whereas the SVM parameter \( C \) ranged from 1e-10 to 10000 in powers of 10 increments. At each iteration (for a given set of parameters), the training set was run through a 10-fold stratified cross-validation. The receiver operating characteristic (ROC) score is used for optimization. By iterating through the parameters, the model optimized for ROC score is obtained, and is subsequently used to test unseen (validation set) data. The output of the model includes a list of feature \( k \)-mers and weights associated with each feature \( k \)-mer that was used to build the model. The feature \( k \)-mers are the support vectors that were used by the SVM to distinguish between the two classes. The value of the feature weight depicts the significance of the feature.

\footnotetext[1]{version PyML-0.7.13.3, Python version 2.7 [21] \texttt{http://pyml.sourceforge.net}}
4.2.4 Discriminating between HuR and TTP binding sites using a Combined Model

To dissect the features both common and specific to the motifs, the datasets (Section PAR-CLIP HuR and TTP clusters) were partitioned, using bedtools [145], into three subsets: Data-set A contained sequence clusters that bound HuR but not TTP (“HuR only”), Data-set B contained sequence clusters that bound TTP but not HuR (“TTP only”), and Data-set C contained clusters that were common to both HuR and TTP.

The sequences associated with each binding cluster were used for motif discovery. Four different scenarios, as listed in Table 4.6, were utilized to discriminate between the HuR and TTP datasets. Using the $k$-spectrum kernel, a HuR specific model $M_H$, TTP specific model $M_T$, and a combined model $M_B$ using clusters common to both proteins were built by varying the parameters $C$, $K1$, $K2$ and optimized for ROC.

4.2.5 Discriminating between HuR and TTP binding sites: Using Multi-task Learning and Domain Adaptation

To identify the shared motifs between HuR and TTP, we utilized Multi-task Learning (MTL) as this approach leverages both the commonality and differences between the RBPs. To identify features specific and shared by domains, Daume III [40] proposed feature augmentation approach (a domain adaptation technique) wherein the feature space is tripled (when the number of domains=2). For each feature in the original feature space, there is a “shared” version and two domain-specific versions, which in our case are “HuR-specific” and “TTP-specific”. This is illustrated in Figure 4.1.

The underlying algorithm is then expected to learn which features transfer across domains (hence shared features) and which do not (domain specific features). Domain adaptation selects features that are either domain specific (HuR or TTP specific) or shared in order to best explain the data. When a domain (HuR or TTP) specific feature is chosen, it implies that the sequence
Listing 4.1: Algorithm to build k-spectrum kernel model

```python
bestROC = -numpy.Inf
K1 = [8, 9, 10, 11, 12, 13]
K2 = [8, 9, 10, 11, 12, 13]
Cs = [10**x for x in xrange(-10, 5)]
for k1 in K1:
    for k2 in [x for x in K2 if x >= k1]:
        for C in Cs:
            trainData = get_spectrum_data(trainSeqFasta, k1, k2)
            s = svm.SVM(C=C)
            s.train(trainData)
            results = s.stratifiedCV(trainData, numFolds=10);
            roc = results.getROC();
            if roc > bestROC:
                trainData.save(dataFile);
                s.save(modelFile)
                bestROC = roc;
                bestC = C;
                bestK1 = k1;
                bestK2 = k2;
bestC, bestK1, bestK2 are obtained here.
```
can be discriminated based on the domain specific knowledge, whereas when a shared feature is selected, the feature is shared between the domains. Each RBP constitutes a domain.

In order to run the $k$-spectrum kernel using domain adaptation, we constructed HuR ($M_H$) and TTP ($M_T$) models independently using 80% of the sequences for HuR and TTP. It is important to note that for this experiment, each model was generated from the original dataset; $M_H$ was generated using all the HuR clusters, and similarly $M_T$ from all the TTP clusters described in Section PAR-CLIP HuR and TTP clusters. The features identified by each model independently serve both as potential combined features and domain-specific features in the combined model. The domain specific features are prefixed with the RBP label (“HuR_“ for features obtained from the HuR model, and “TTP_“ for features obtained from the TTP model). The resulting combined model ($M_B$) then identifies the most discriminative set of features that explain each domain from the combined features and domain specific sets.
4.3 Results

4.3.1 Predict RBP binding sites using the k-spectrum kernel

We first sought to determine whether the k-spectrum kernel could accurately and independently capture the known motifs without utilizing the cross-linking information. Prior work on identifying HuR motifs [100, 130, 149] utilized exclusively on the identification of 7-mers around the cross-linking site, identified by the alteration induced in the sequence. We utilized the k-spectrum kernel method on each RBP (HuR and TTP) independently leveraging only the cluster data which ignores the sequence change induced by the cross-linking. For both RBPs, we calculated the success rate of our k-spectrum model on held-out test data.

We first note that the k-mers used by the model to discriminate HuR clusters from the control data were consistent with the published HuR motifs (see Table 4.2). The HuR model classified the test data with a 77.3% success rate (see Table 4.2). Additionally, after U-rich k-mers the second highest scoring feature was AU-rich k-mers that correspond to the motifs identified by the miReduce [171] algorithm used by Lebedeva et al. [100]. Interestingly, the k-mers were generally longer (9 to 12 nucleotides) than the published 7-mers. Our results suggest that the published motif was flanked by U’s, consistent with the U-rich nature of HuR binding sites.

We next sought to compare the k-spectrum kernel [102] results on HuR to the discriminative motif finder DREME [13]. The top motifs identified by DREME [13] were of length 7 (best E-value from motifs of length 7 to 13). The top DREME [13] motif, HUUUUHW, was found in 2,871 out of 3,642 sequences and had an E-value of 6.6e-143 (See Figure 4.2 for associated sequence logo.) The over-represented k-mers associated with this motif are listed in the Table 4.2. Both discriminative methods, the k-spectrum kernel method and DREME, discovered HuR motifs that were consistent with the published motifs, though the k-spectrum kernel method returned a longer motif.

We utilized a similar approach to validate the k-spectrum kernel on the prediction of TTP binding motifs. The k-spectrum kernel method was reasonably accurate, classifying the test data with a 88.8% accuracy rate. The top scoring features were AU-rich 9-mers, including the well
<table>
<thead>
<tr>
<th>Method</th>
<th>Success Rate</th>
<th>Sensitivity</th>
<th>PPV</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-spectrum</td>
<td>77.3</td>
<td>96.8</td>
<td>69.7</td>
<td>89.4</td>
</tr>
<tr>
<td>DREME</td>
<td>78.6</td>
<td>92.5</td>
<td>64.1</td>
<td>87.3</td>
</tr>
</tbody>
</table>

Table 4.2: Discriminative methods recover known K-mers for HuR. Top feature k-mers from the k-spectrum kernel and over-represented k-mers from DREME are compared to the published k-mers [100]. For the k-spectrum kernel, determined feature weights are provided.

![Position Frequency Matrix for the top motif identified by DREME for HuR](image)

Figure 4.2: Position Frequency Matrix for the top motif identified by DREME for HuR

known nonamer AUUUAUUUA (type 2 RRE) [14] that is specific to TTP binding targets [25]. DREME also identified AU-rich motifs, but shorter in length (7 nt) with the top DREME motif,
ATANWTW scoring an E-value of $8.7 \cdot 10^{-743}$. This motif was found in 2,871 out of the 4,626 sequences. See Figure 4.3 for associated sequence logo. Table 4.3 compares the $k$-mers found by the $k$-spectrum kernel [102], DREME [13], and the published TTP motifs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Success Rate</th>
<th>Sensitivity</th>
<th>PPV</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-spectrum</td>
<td>88.8</td>
<td>95.5</td>
<td>84.1</td>
<td>95.6</td>
</tr>
<tr>
<td>DREME</td>
<td>91.4</td>
<td>91.37</td>
<td>77.39</td>
<td>94.4</td>
</tr>
</tbody>
</table>

**Published Motifs [130]**

<table>
<thead>
<tr>
<th>$k$-spectrum Features and Weights</th>
<th>DREME over-represented $k$-mers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUAAUUAU</td>
<td>UAAAUAAU. 6.44</td>
</tr>
<tr>
<td>AUAAUUA</td>
<td>UAUAUAAUAU. 5.4</td>
</tr>
<tr>
<td>AUAAUAU</td>
<td>AUAAUAUAU. 5.34</td>
</tr>
<tr>
<td>UUAAU</td>
<td>AUAAUAUAAUAU. 5.17</td>
</tr>
<tr>
<td>AUUUAU</td>
<td>AUAAUAUAAU. 4.72</td>
</tr>
</tbody>
</table>

**Table 4.3:** Discriminative methods recover known K-mers for TTP. Top feature $k$-mers from the $k$-spectrum kernel and over-represented $k$-mers from DREME are compared to the published $k$-mers [130]. For the $k$-spectrum kernel, determined feature weights are provided.

![Figure 4.3: Position Frequency Matrix for the top motif identified by DREME for TTP](image)

We next sought to determine whether we could improve on these models using feature engineering. Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy.
on unseen data. As a great deal is already known about both HuR and TTP, we wondered if the addition of known features, namely that HuR binds U-rich and TTP binds to AU-rich regions, would improve the performance of our $k$-spectrum kernel models. The engineered features, normalized by sequence length, were:

- 4 features: Number of As, Us, Gs or Cs found in each sequence (aCount, uCount, gCount, cCount).
- 2 features: Number of contiguous As or Us found in each sequence (aRepeatedCount, uRepeatedCount).
- Total number of AU-rich dimers (AA, AU, UA, UU) found in each sequences (auCount).

The $k$-spectrum kernel method [102] was run using these additional features on both the HuR and TTP datasets and evaluated against the same parameters as the models developed without feature engineering. Table 4.4 lists the results of incorporating feature engineering into the $k$-spectrum kernel method.

<table>
<thead>
<tr>
<th>RBP</th>
<th>Success Rate</th>
<th>Sensitivity</th>
<th>PPV</th>
<th>ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HuR</td>
<td>82.4</td>
<td>86.4</td>
<td>79.7</td>
<td>89.7</td>
</tr>
<tr>
<td>TTP</td>
<td>85.2</td>
<td>91.2</td>
<td>81.5</td>
<td>92.6</td>
</tr>
</tbody>
</table>

Table 4.4: Feature engineering does not consistently improve model performance. Performance (success rate, sensitivity, PPV and ROC) for models for HuR and TTP incorporating engineered features.

The engineered features topped the list for both proteins, suggesting that the generalized binding preferences are high-quality descriptions of the binding site preferences of these RBPs. Interestingly, the top two features (uCount and auCount) were identical for both HuR and TTP, only in different orders, suggesting that both proteins indeed have similar binding preferences. The relative similarity of scores for these features on TTP further suggests that TTP binds to both AU-rich and U-rich sites somewhat equivalently. Table 4.5 enumerates the high scoring features for both HuR and TTP after the addition of engineered features. It is interesting to note, however, that inclusion of these engineered features did not consistently improve the performance of the model...
(See Table 4.4), indicating that the \( k \)-mers identified adequately recover the generalized feature descriptors.

<table>
<thead>
<tr>
<th>HuR (Feature, Weight)</th>
<th>TTP (Feature, Weight)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uCount, 1142.8</td>
<td>auCount, 1488.15</td>
</tr>
<tr>
<td>auCount, 956.9</td>
<td>uCount, 1462.98</td>
</tr>
<tr>
<td>uRepeated, 671.43</td>
<td>uRepeated, 894.11</td>
</tr>
<tr>
<td>UUUUUUUU, 46.69</td>
<td>aCount, 337.70</td>
</tr>
<tr>
<td>UUUUUUUU, 30.21</td>
<td>UUAUUUAUU, 38.17</td>
</tr>
<tr>
<td>AUUUUUUU, 19.88</td>
<td>UAUUUAUUU, 32.69</td>
</tr>
<tr>
<td>UUUUAUUU, 19.64</td>
<td>AAUAUUUAU, 26.33</td>
</tr>
<tr>
<td>UUUUUCUU, 17.99</td>
<td>AAUAUUUAU, 25.83</td>
</tr>
<tr>
<td>UUUUCUUU, 16.68</td>
<td>UUAUUUAUU, 20.89</td>
</tr>
<tr>
<td>UUUUUUUU, 15.63</td>
<td>AUAUUUAUA, 20.22</td>
</tr>
</tbody>
</table>

Table 4.5: Engineered features obtain the top weights in the model. Top ten features along with their weights discovered by the \( k \)-spectrum kernel method when feature engineering is incorporated.

4.3.2 Discrimination between HuR and TTP using \( k \)-spectrum models

Since both the proteins had similar high-scoring features we next sought to determine whether the models could discriminate between the binding sites of HuR and TTP. Hence, we sought to determine whether it could identify \( k \)-mers that distinguish between HuR and TTP binding. To this end, we utilized the three categories of peaks (HuR only, both HuR and TTP, and TTP only) in different scenario combinations (see Table 4.6) as both positive and negative sets. We reasoned that peaks bound only by HuR may harbor features distinct to its binding relative to TTP, and vice versa. For each case, we utilized both the \( k \)-spectrum kernel and DREME for analysis.

First, we sought to identify the features key to the HuR only dataset (the positive) distinct from TTP and common clusters (scenario 1 of Table 4.6). The resulting \( k \)-spectrum model identified U-rich features as the most discriminative. As shown in Table 4.7, it was able to discern between the two sets with a 79% balanced success rate. DREME identified over-represented \( k \)-mers were similar to those identified by the \( k \)-spectrum kernel but typically shorter in length (Table 4.7), a pattern that was consistent across all four scenarios. Therefore, we focus our subsequent discussion on the \( k \)-spectrum results.
To identify features specific to the TTP dataset, we employed TTP only as the positive and all of the HuR and common clusters as the negative dataset (scenario 2 in Table 4.6). The discriminating features were AU-rich, with several containing one or more overlapping repeats of AUUUA. Further analysis of this repeating pentamer revealed that it was found 206 times in the TTP only sequences, only 25 times in the HuR only sequences, and 11 times in the common sequences suggesting that it is indeed a discriminating feature for TTP binding.

We next sought to identify the features that were typical of commonly bound regions (scenario 3 in Table 4.6). The model identified AU-rich features, though they had appreciable fewer A’s than those identified for TTP only. This suggests that $k$-mers intermediate between the U-rich features of HuR and the AU-rich features of TTP can be recognized by either protein. The success rate of the model, however, was discernibly lower (by 10-15 percentage points) than the HuR only and TTP only models. In line with the drop in accuracy, we noted that most instances of the top $k$-mers were found in the HuR only (1668 instances) and TTP only (1542 instances) datasets rather than the combined set (324 instances) (See Figure 4.4).

Finally, we wondered whether the inclusion of these common events was reducing the accuracy of our HuR specific and TTP specific models. To test this, we ran one last scenario, namely where the HuR only is used as a positive set and the TTP only as a negative (scenario 4 in Table 4.6). The resulting feature set qualitatively resembled those found for earlier HuR models, both the discriminative one (scenario 1 in Table 4.6) and the HuR independent model (Table 4.2), but the success rate improved a little relative to both models. While the increase in accuracy is small (2-3%), it does suggest that HuR binds a distinctly more U-rich motif than TTP and that discrimination

<table>
<thead>
<tr>
<th>Test</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>Only HuR</td>
<td>Only TTP + (HuR And TTP)</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Only TTP</td>
<td>Only HuR + (HuR And TTP)</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>(HuR And TTP)</td>
<td>Only HuR + Only TTP</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>Only HuR</td>
<td>Only TTP</td>
</tr>
</tbody>
</table>

Table 4.6: Four different test scenarios to identify motifs shared and specific to each RBP.
by sequence features is possible.

**Figure 4.4: Analysis of Feature k-mers:** (A) Number of occurrences of HuR specific feature k-mers in each dataset. (B) Number of occurrences of TTP specific k-mers in each dataset. (C) Number of occurrences of common k-mers in each dataset.

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only HuR vs (Only TTP and Common)</td>
<td>Only TTP vs (Only HuR and Common)</td>
<td>Common vs (HuR and TTP specific)</td>
<td>HuR-specific vs TTP-specific</td>
</tr>
<tr>
<td>Positive Sequences</td>
<td>1206</td>
<td>1011</td>
<td>904</td>
</tr>
<tr>
<td>Negative Sequences</td>
<td>5657</td>
<td>6142</td>
<td>7987</td>
</tr>
<tr>
<td>Balanced Success Rate</td>
<td>78.1</td>
<td>72.8</td>
<td>62.9</td>
</tr>
<tr>
<td>ROC</td>
<td>85.2</td>
<td>80.5</td>
<td>68.7</td>
</tr>
<tr>
<td>k-spectrum Features</td>
<td>UUUUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCUUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCUUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUU</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td>DREME Motifs</td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
<tr>
<td></td>
<td>UCCCCUUC</td>
<td>UACUAUUCA</td>
<td>UCCUUCAUAA</td>
</tr>
</tbody>
</table>

**Table 4.7:** Discriminative methods evaluate distinct subsets of the HuR and TTP dataset to identify both shared and specific sequence features. The performance of the k-spectrum kernel on distinct subsets of the data. Top eight k-mers from k-spectrum kernel (by weight) and DREME (by E-value). The test scenarios correspond to Table 4.6.

### 4.3.3 Discrimination between HuR and TTP using Multi-task learning and Domain Adaptation Technique

Given the limitations of a pure k-spectrum kernel approach at identifying features typical of commonly bound regions (results for scenario 3 in Table 4.6), we next employed the feature augmentation technique in domain adaptation along with multi-task learning to identify not only the features specific to each domain but also those that are shared by the domains. We combined features with domain information to construct a combined model $M_B$. We then compare this model to the domain specific models $M_H$ (HuR) and $M_T$ (TTP).
First, we conducted an out-of-domain test on $M_H$ and $M_T$ where the test data belonged to a different domain than the model. As seen in Table 4.8, the results show that the HuR model was able to discriminate TTP binding sites with a higher ROC score than the TTP model on HuR data. However, given our feature engineering results, it is not possible from this data alone to distinguish whether this indicates that the HuR dataset is simply more diverse (e.g. HuR binds a broader range of targets) or whether the TTP protein is actually more discriminative.

Then the combined model was run against each domain data as well as the combined dataset. The intuition behind the combined model is that it should predict HuR, TTP, and combined data with at least the same ROC score as the individual models since it contains the original set of features of both domains. As seen in the Table 4.8, the combined model $M_B$ had the same ROC score for HuR dataset, and a slightly lower (less than 1%) ROC score for TTP data set. This suggests that the combined model may have a slight HuR bias, possibly stemming from the fact that HuR targets are typically longer in length. Consistent with this possibility, the combined model’s top features were HuR domain specific.

Interestingly, several of the domain specific features are also identified as shared. For example, HuR_UUUUUUUUU and UUUUUUUUU are identified as the second and third top $k$-mers. This indicates that poly-U tracks are recognized by both RBPs, hence it is a shared feature, but that it has higher significance in the HuR domain. In fact, all domain specific $k$-mers were also identified as shared except the last one, e.g. TTP_AUUAUUUA, which corresponds to the known TTP specific nonamer. This is consistent with the feature identified by the TTP model (See Table 4.3). Despite the fact that only one $k$-mer (in the top twenty $k$-mers) is uniquely domains specific, the combined model ROC score was 92.5%, a slight improvement over the out-domain capabilities of the HuR only model in predicting TTP. This suggests that the combined model does identify distinguishing features for TTP that contribute to its success, despite the fact that they are not in the top ten features.
<table>
<thead>
<tr>
<th>Test Data/Model</th>
<th>HuR Model ($M_H$)</th>
<th>TTP Model ($M_T$)</th>
<th>Combined Model ($M_B$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HuR</td>
<td>89.4</td>
<td>87</td>
<td>89.4</td>
</tr>
<tr>
<td>TTP</td>
<td>91.2</td>
<td>95.6</td>
<td>94.5</td>
</tr>
<tr>
<td>Combined</td>
<td>NA</td>
<td>NA</td>
<td>92.5</td>
</tr>
</tbody>
</table>

### Features

- **HuR**
  - UUUUUUUU UAUUUAAAU
  - UUUUAUUUU UAAUUAAAU
  - UUUCUUCU UUAUUAUUA
  - UUAUAUUU AAUAAAUUA
  - AUUUUUUU UAAUUAAU
  - UUUUCUUU AUAUUAAU
  - UUUUAUUU CUAAUAUAA
  - UUUUUUCU AUAUUAAU
  - UUUUUUUC ACUUUUAUU
  - UUAUUUUU UCUAAUAUA
  - UUUUUUAA UAAUUUAAC
  - UUUUUUUC UAAUUAAU
  - UUUUUUUC UUAUUAUUA
  - UUUUUUUC UUUUUUAU
  - UUUUUUUC UCUUUUAUU
  - UUUUUUUC UUUUUUAU
  - UUUUUUUC UUUUUUAU

- **TTP**
  - UUUUUUUU UAUUUAAAU
  - UUUUAUUUU UAAUUAAAU
  - UUUCUUCU UUAUUAUUA
  - UUAUAUUU AAUAAAUUA
  - AUUUUUUU UAAUUAAU
  - UUUUCUUU AUAUUAAU
  - UUUUAUUU CUAAUAUAA
  - UUUUUUCU AUAUUAAU
  - UUUUUUUC ACUUUUAUU
  - UUAUUUUU UCUAAUAUA
  - UUUUUUAA UAAUUUAAC
  - UUUUUUUC UAAUUAAU
  - UUUUUUUC UUAUUAUUA
  - UUUUUUUC UUUUUUAU
  - UUUUUUUC UCUUUUAUU
  - UUUUUUUC UUUUUUAU
  - UUUUUUUC UUUUUUAU

**Table 4.8:** Domain adaptation and multi-task learning identifies domain specific (prefaced with HuR or TTP) and shared (no prefix) k-mers. Performance metrics and top twenty k-mers are compared for the HuR, TTP and combined model (see main text for description of models).

### 4.4 Discussion

Two discriminative methods, the $k$-spectrum kernel method [102] and DREME [13], both discovered HuR and TTP motifs that were consistent with the published motifs; the HuR $k$-mers were predominantly U-rich, and AU-rich for TTP. While the success rate of these methods was comparable, the $k$-spectrum kernel method had higher sensitivity and PPV values than DREME. With discriminative methods, sensitivity and specificity are often trade-offs. This is likely the case here, as DREME had a slightly better success rate. In some cases, a higher sensitivity is preferred,
particularly when subsequent experiments will validate the predicted sites and there is a cost to missing potential targets.

Additionally, relative to DREME [13], the $k$-spectrum kernel method identified slightly longer $k$-mers that provide additional insight into the flanking regions around the core motif. HuR is known to bind its targets using two RNA binding domains. Wang et. al. [190] found that HuR binding to $c$-fos RNA involved an 11-base segment 5´-AUUUUAUUUU-3´, in support of a longer recognition sequence. Given the success of the $k$-spectrum kernel method at identifying binding motifs that matched known published motif of HuR and TTP, this method could be potentially used to identify de novo binding sites of other RBPs.

The case of HuR and TTP binding is particularly paradoxical, as both proteins recognize similar sequence features. In fact, feature engineering did not consistently improve the accuracy with which the $k$-spectrum kernel discriminated between RBP set and the control set. However, the engineered features did bubble up to be highly discriminative, suggesting that the binding sites of HuR and TTP are indeed simply U/AU-rich. Likewise, the features identified as domain specific using multi-task learning were often also shared features (such as HuR_UUUUUUUU and UU-UUUUUUU, HuR_UUUUAUUUU and UUUUAUUUU) suggesting that both proteins are capable of recognizing these $k$-mers but with distinct affinities. These results both confirm and provide further evidence for the hypothesis that these RBPs do indeed share similar binding preferences and varying affinity shifts decide which RBP would bind to a particular target.

Not all RNA binding proteins utilize primary sequence for recognition. Instead, some RNA binding proteins may recognize secondary structure. The co-variation inherent in conserved secondary structures is significantly more difficult to detect than primary sequence. It is unclear how well a $k$-spectrum kernel approach would fare when the target motif is structural. The next chapter discusses a framework developed in Python that benchmarks computational methods by varying the target motif and other conditions.
RBPs are known to recognize both sequence-specific and structural targets. The field of RBPs is growing, and there are several computational methods being developed to identify their targets. However, given the RBP binding preferences, it is hard to predict which method would be suitable for a particular RBP being studied. This chapter describes the architecture I developed in Python to benchmark computational methods. The framework I developed called the PySG first generates datasets by embedding user-defined motifs (primary and secondary structure) followed by benchmarking the performance of computational methods.

The architecture of this framework is described in Section 5.3 which details the different configuration parameters to generate datasets, the process of creation of positive and negative sequences, running computational tools, comparing the results of the computational tool with the embedded signals, and finally benchmark the performance of computational tools. While the initial design of the framework involves utilizing a base set of sequences to generate datasets, ultimately the goal is to generate these datasets using a generative language model. Augmenting the PySG framework to support additional types of signals as well as benchmark another computational method is described in Section 5.4. The chapter concludes by outlining the future work for this framework.
5.1 Motivation

Evaluating the efficacy of computational tools first requires knowing the solution, which is tricky in the case of tools focused on de-novo identification of binding targets or motifs. RNA binding targets can recognize secondary structure in addition to a primary sequence which further makes the discovery of RBP targets harder. Secondary targets can be falsely identified as seen in the case of HuR targets identified by RNaPromo [147] (See Section 2.5.2 from Chapter 2) or missed when the focus of computational tools is primary sequence motifs. RBPs such as HuR are known to bind to longer mRNAs, whereas TTP is known to bind to shorter mRNA targets [130]. Cluster sequences from PAR-CLIP [72] experiments are known to have sequencing errors or other anomalies that would result in the RBP motif to be absent in some clusters but present in others. Hence, manipulating sequence length and “signal percentage” are also becoming key biological conditions that need to be simulated by sequence generators (in this chapter, and next, the signal will refer to the specific type of motif or secondary structure). Hence, there is a need for a framework that can:

- Allow users to specific experimental configuration parameters that will drive the generation of sequences.
- Simulate sequences that follow a particular nucleotide distribution (e.g. 3´UTRs or 5´UTRs).
- Generate control set (negative sequences) by shuffling the positive set while keeping di/trinucleotide distribution constant.
- Allows users to embed specific types of “signals” in the generated sequences.
- Allow users to configure the number of “signals” as well as the location where the signal is embedded in the generated sequences.
- Benchmarks computational tools based on configuration parameters. Adding new types of signal and computational tools should be a relatively simple task to perform.
• Ability to benchmark individual parameters or perform a grid search to find optimal parameters for each computational tool.

A review of existing benchmarking tools listed by National Institute of Health (NIH) [137] revealed that the existing simulators primarily simulated sequences in the field of genomic evolution, phylogenetics, next generation sequencing (NGS), mutations (Single Nucleotide Variation, Insertion, and Deletion, Inversion and Rearrangement, Genotype or Sequencing Error) (See Appendix [E] for more details). One particular tool in the NGS category, Benchmarker for Evaluating the Effectiveness of RNA-Seq Software (BEERS) [69] was closest to the requirements listed. However, the focus of BEERS [69] was to simulate RNA-Seq data from either the mouse, and human genome by using annotations from ten different published annotation data, selects a fixed number of genes at random, and then introduces mutations, and sequencing errors to generate a set of reads. The generated reads are then benchmarked to measure the accuracy and robustness of available RNA sequencing (RNA-Seq) algorithms. While this is also a benchmarking tool, the scope of this tool is restricted to generating reads to simulate the output of the different sequencers (Illumina, Roche, etc.). It does not intend to create sequences of varying lengths or allow users to specify the type of motif to embed in the sequences. A lot is known about RBPs and their binding characteristics, but there aren’t frameworks that simulate their binding preferences. Therefore, I developed a framework that would generate sequences that would be characteristic of what RBPs recognize followed by benchmarking computational methods on the created datasets. The characteristics of RBP binding is discussed next section.

5.2 Characteristics of RBPs

On a very high-level, the RBPs exhibit a modular structure and consists of multiple binding motifs that are built from a small number of basic domains. These domains also known as RNA binding domains (RBDs) dictate the nature of binding and are discussed in Chapter 2, Section 2.2.1. Figure 5.4 demonstrates the different binding mechanisms of RBPs. The Pfam database [56]
contains multiple sequence alignments of protein families that are generated using hidden Markov models (HMMs). Pfam domains group RBDs based on sequence specificity; for example, the RRM-like clan contains families that are related to the RNA recognition motif (RRM) domains. While they have similar binding preferences, not all the families bind RBPs, and for some, while the binding is known, the binding motif or function is unknown (e.g. Domain of unknown function (DUF)).

RBDs are several nucleotides long e.g. RRM domain typically is 80-90 nucleotides long, but not all of those nucleotides take part in the binding (contact point between the RBP and the mRNA). The contact can be made directly with the RNA bases, the ribose sugar, and phosphate groups. In the case of some single-stranded RNA (ssRNA) binding domains such as RNA recog-
ntion motif domain (RRM), this interaction is through four to eight nucleotides. The specificity (strength of binding between protein and nucleotides) in this binding is mainly through hydrogen bonds and van der Waals interaction of the nucleotides with the protein side chains keeping the sugar-phosphate backbone exposed to the solvent [11]. This domain consists of conserved basic and aromatic residues. The basic residues produce a positively charged surface on the RRM which facilitates RNA binding whereas the aromatic residues help with the base-stacking interactions with the target RNA [112]. The binding motif thus could be spread across nucleotides in this domain and the contact points could be spread across the sequences thus resulting in non-contagious motifs. RRM contains two highly conserved motifs: Ribonucleoprotein (RNP)1 and RNP2. The amino acids in the RNP are involved in the interaction with the target RNA. When the RRM and the target RNA interact, the ssRNA target sequence lies across the surface of the β sheet in the RRM and the amino acids in the RNP1 and RNP2 interact with the RNA target sequence. The presence of multiple RRMs with sequences in between them, adds to the differences in the recognition of target sequences and binding mechanisms of RBPs. The sequences in between the RRMs are also known as linker sequences. The location of contacts made between the protein and RNA include the amino acids in the RNP1 and RNP2 and are the motifs involved in the binding (See Figure 5.5).

The K-Homology domain is known to bind to both ssRNA and double-stranded RNA (dsRNA) and is characterized by a hydrophobic core domain, and typically recognizes four nucleotides on ssRNA sequences through electrostatic interactions [184]. The KH proteins are bind through a combination of RBDs; for example, the IGF2Bp1 family is known to contain four KH and two RRM motifs in its binding [18]. The proteins in the zinc-finger domain form sequence-specific interaction with the RNA through hydrogen bonding and van der Waals interactions of the protein backbone [112]. The affinity of this binding is increased by using stacking interactions of aromatic side chains of the protein with the RNA bases. The rigidity and shape of the protein structure are key determining factors for the specificity of zinc-finger proteins to target RNAs. Like the KH domains, the zinc finger domains also occur in repeats or co-bind with other RBDs [18]. The
ZFP36 family is known to bind to the zinc-finger domain. Following the binding, each of the two zinc fingers formed similar structures and the contact to the mRNA exhibited the 5′-UAUU-3′ also known as the half-site. Typically, the ZFP protein families bind to the 5′-UUAUUUAUU-3′ nonamer. Thus, identifying binding motifs when there are structural elements involved is a harder problem to solve. Figure 5.6 demonstrates the binding of human TTP TZF domain to the nonamer sequence UUAUUUAUU. The structure is based on the ZFP36L2 bound to the same sequence. The two zinc fingers are seen to the left and right and the RNA nucleotides run from left to the right in the 5′-3′ orientation.

Finally, the double-stranded RNA (dsRNA) binding domains achieve specificity through recognition of the shape of RNA secondary structure such as stem-loops by interacting with dsRNA. The dsRBD is one of the most abundant RNA binding domain after RRM and zinc-finger domains. The conformation of dsRNA is the A-form RNA double helix. In this helix, there is a wide and shallow minor groove and deep and narrow major groove. The access to the bases is readily available in the minor groove whereas it is hindered in the major groove. In addition, the presence
of 2’ OH functional groups that line the minor groove is distinct in dsRNA when compared to DNA double helix. The contacts in such binding occur between RNA base and dsRBDs in the minor groove; a feature specific of dsRBDs [119] (See Figure 5.7).

The regions where RBPs bind is another characteristic of RBPs. Polypyrimidine tract binding protein (PTB) are known to bind to the pyrimidine-rich tract of the 3’ end of the introns. Hu proteins typically bind to the AU-rich elements (AREs) that are often present in the 3’ UTR of mRNAs. The Puf family of proteins are known to bind close to the end of the 3’ UTR as well around the poly-A tail [125]. Thus, the location of binding is another key characteristic of RBP binding.
5.3 Sequence Generation Architecture

The framework for PySG (developed in Python) is a multi-step process as outlined in the Figure 5.8. The PySG creates FASTA sequences, runs these sequences through various computational tools, benchmarks the performance of the computational tools under different experimental
configurations by graphing the results of each computational tool. Each step of the architecture is described as its own step.

5.3.1 Step 1: Create Experimental Configuration

To create sequences under varied experimental conditions, it is necessary to have a configuration file that would allow these conditions to be set in a simple, user-friendly, and efficient manner. In the PySG framework, this is achieved by creating a driver configuration file in the YAML Ain’t Markup Language (YAML) format (See Example 5.1 below). The YAML format makes the configuration file human-readable thus user-friendly, and the parameter names are intuitive. The choice of configuration file format was selected as Python has a PyYAML library that parses the config-
uration file and is fed downstream to the application. Integration with this library was relatively simple.

The experiment configuration file specifies different parameters under which to create FASTA sequences and run computational tools. This is the only step that requires manual/user intervention. Once the driver configuration file is created, the rest of the framework is completely automated.

Listing 5.1: Experiment Configuration example

targetDir: 'Fus_Test_SeqLength_Shuffle'
numSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
  type: 'pwmFiles'
pwmFiles: ['FusIP_1.pwm']
signalPercent: [75]
locationFromStart: [10]

nosignal:
  type: 'shuffle'
fastaFile: '3UTR_transcripts_Human.txt'

The parameters that can be configured are listed in Table 5.1. The configuration parameters are divided into two types: global parameters which apply to both the set of sequences (positive, and control), and signal parameters that are used to specify the type of signal, location of the signal, and the number of sequences that contain the signal. The file formats supported to represent the signal are position weight matrix (PWM), position frequency matrix (PFM), IUPAC motif, FASTA format of multiple sequence alignment of secondary structure, and specific k-mers.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetDir</td>
<td>Valid path</td>
<td>Directory to store results</td>
</tr>
<tr>
<td>numSeq</td>
<td>[Any positive number]</td>
<td>Number of sequences to create</td>
</tr>
<tr>
<td>seqLen</td>
<td>[Any positive number]</td>
<td>Sequence lengths to create</td>
</tr>
<tr>
<td>signal:type</td>
<td>PWM, PFM, Structure, IUPAC motif</td>
<td>Signal type to embed</td>
</tr>
<tr>
<td>nosignal:type</td>
<td>Shuffle</td>
<td>FASTA sequence set</td>
</tr>
<tr>
<td>fastaFile</td>
<td>FASTA file</td>
<td>FASTA file containing a set of sequences</td>
</tr>
<tr>
<td>locationFromStart</td>
<td>[Positive Number &lt; Sequence Length]</td>
<td>Location from start to embed motif</td>
</tr>
<tr>
<td>locationFromEnd</td>
<td>[Positive Number &lt; Sequence Length]</td>
<td>Location from end to embed motif</td>
</tr>
<tr>
<td>signalPercent</td>
<td>[0-100]</td>
<td>Percent of sequences containing the motif</td>
</tr>
</tbody>
</table>

Table 5.1: Configuration Options to setup benchmarking parameters
To evaluate the effect of each individual parameter, the experiment configuration allows values to test to be specified as a list. For example, to evaluate the performance of the computational tools on varying sequence lengths, the seqLen parameter can be set to $[128, 256, 512, 1024]$. If all other parameters are single values, then the experiment configuration will generate four individual configuration files. The targetDirectory is the location where all the configuration files, FASTA files are housed. The seqLen parameter is a list of sequence lengths to be generated. The numSeq is a list that contains the number of sequences to be generated. The fastaFile is a FASTA file from which sequences are generated if the shuffle option is selected to generate sequences (More details in Section 5.3.3). The location of the motif to embed is specified using either locationFromStart or locationFromEnd parameters. The signalPercent parameter configures the percentage of the sequences that contain the motif. This is an especially important parameter as this simulates the absence of the motif in sequences, and tests the discriminative method.

### 5.3.2 Step 2: Generate Individual Configuration Files

The goal of this step is to accept the driver configuration file specified by the user in Step 1 and generate individual “run” configuration files. This is done by iterating through the different configurable parameters in the driver configuration file. The beauty of this step is that the individual experiment configuration files are generated programmatically based on the configuration specified in Step 1.

The example specified in Section 5.3.1 will result into four individual experiment configuration files as it will iterate through the only parameter which is specified as a list: seqLen. Since all the other parameters have a single value, only four configuration files are generated. An example of one configuration file that resulted from the experiment configuration is:

```
sequence:
global:
  fastaFile: 3UTR_5UTR_Sequences.fa
  numSeq: 512
  outNoSignalFastaFile: NoSignal_512_128_75_FusIP_1.pfm.fa
```

**Listing 5.2:** Individual configuration file
The configuration file follows the YAML format and contains two sub-sections “global”, and signal. The “global” section contains the fastaFile which is the name of the FASTA file that contains a set of FASTA formatted sequences used to generate sequences for each experiment. In this example, that file is a set of 3’UTRs and 5’UTRs transcript sequences. The parameter numSeq and seqLen specifies the number of sequences, and the length of the sequences to generate respectively. The outNoSignalFastaFile parameter specifies the name of the control FASTA file to generate. The name of this file is generated programmatically in Step 1 when the individual configuration files are written. The name is delimited by “_” and the “NoSignal” refers to control set, “512” refers to the number of sequences generated, “128” refers to the sequence length, “75” refers to the number of sequences that will contain the motif, and “FusIP_1.pfm” is the motif for the RBP Fus in the Position Frequency Matrix (PFM) format. The positive filename is constructed in a similar fashion except the prefix contains Signal instead of NoSignal.

The “signal” section specifies the locationFromStart parameter which is the location in the sequence the signal is embedded, the outSignalFile which specifies the FASTA file name to save the generated positive set of sequences, the pfmFile which specifies the PFM file to use to generate the signal, and finally signalPercentage which sets the percentage of the sequences that contain the signal. The default value of signalPercentage is 100% when not specified in the configuration file. In this example, 75 % of the sequences will contain the motif.

The output of this step is a list of individual experiment configurations generated and saved in the “targetDir” parameter specified in the driver configuration file (Step 1).
5.3.3 Step 3: Creating FASTA files

The objective of this step is to create FASTA sequences for each experiment using the individual experiment configuration generated in Step 2. Chapter 3 Section 5.2 discussed the use of generative models to create synthetic datasets. However, for the first revision of the PySG framework, we wanted to ensure that the datasets created more realistically simulated the characteristics of RBPs and the binding mechanics involved with both single stranded and double stranded RNAs. Figure 5.9 outlines the stepwise procedure of creating these sequences.

Creating Base Sequences: This step is governed by two global configuration parameters: seqLen and numSeq. N sequences of length L nucleotides are created by randomly selecting sequences from the list of 3´UTR and 5´UTR transcript sequences. These transcripts are downloaded from BioMart [167] (hg19 - human genome) and are used to create the background sequences before embedding a particular signal. From the set of transcripts, a random transcript is chosen, and a random location within the transcript is picked, and from that random location, L nucleotides are selected to be one of the created sequences. The selected nucleotides are shuffled to prevent the possibility of any motifs being present in the selected fragment. This process is continued for N sequences. Thus, a dataset consisting of N sequences each of length L is created.

Embedding signal: In order to embed the signal in the generated sequences, the configuration parameters “signal: type” and “signal:signalPercentage” from the signal section of the configuration

Figure 5.9: Process of creating Positive and Control set of sequences
file are looked up. For primary sequence based signals, Tools for Analysis of MOtifs (TAMO) \cite{68} is used to generate the motif. TAMO \cite{68} has been used for interpreting transcriptional regulation using DNA-sequence motifs. TAMO \cite{68} also has inbuilt motif generation and discovery packages and these packages are integrated with the PySG framework. The PySG framework particularly uses the MotifTools package. TAMO \cite{68} accepts PWM, PFM, International Union of Pure and Applied Chemistry (IUPAC) motif representations, and generates probabilistic \(k\)-mers that are representative of the motif. The number of \(k\)-mers generated depends on the \texttt{signalPercentage} value. If this parameter is not specified, \(N\) \(k\)-mers are generated and will be embedded in each sequence generated earlier. For example, if the \texttt{signalPercentage} is 75\% and \(N = 100\), only 75 sequences will be embedded with the \(k\)-mers specified in the configuration file.

To embed secondary-structure signals, the motif to embed is specified in FASTA format and includes multiple sequence alignments of a structured RNA element obtained from Rfam \cite{138}. Rfam \cite{138} is an annotated, open access database that contains information about non-coding RNAs (ncRNAs) as well as other structured RNA elements. Rfam \cite{138} contains multiple sequence alignments (MSA) of ncRNAs and RNA structure elements. The ncRNAs often have similar secondary structure and this is captured into MSAs by Rfam \cite{138}. The MSAs for multiple RNA structure elements that are simple stem-loop structures are saved to the PySG framework. The FASTA file containing the MSAs is listed in the “signal: structureFile” parameter in the configuration file. In order to calculate the number of sequences that contain the signal, the same procedure is followed as done for the primary sequence based signals. The \texttt{signal:signalPercentage} parameter lists the number of sequences that should contain the MSA of the secondary structure element. From the list of MSAs, a random sequence is selected and embedded at the specified location in the configuration file.

Based on the type of the signal (primary or secondary), the output of this step is a set of sequences constructed from selecting random fragments from 3′UTRs and 5′UTRs and embedding them with a specified motif.

\textbf{Control or Negative Set}: To create the negative or control set, each sequence from the positive
set is shuffled but the trinucleotide distribution is consistent with the positive set of sequences. It is important to note that since the positive set is created from fragments of 3′UTRs and 5′UTRs, the nucleotide distribution of the positive set matches that of the 3′UTRs and 5′UTRs sequences. This makes the sequence generation process more realistic rather than “synthetic”. In order to validate the nucleotide composition of the created sequences and 3′UTRs and 5′UTRs sequences, I evaluated the bi/trinucleotide distribution using an $n$-gram model. I computed the 3-gram distribution of the 3′UTR and 5′ UTR transcript sequences and compared that to the created positive and negative set of FASTA sequences (See Figure 5.10). The created sequences match the $n$-gram distribution of the 3′UTR and 5′ UTR transcripts thus validating that the generated sequences closely model the transcripts. This makes the PySG framework different from existing frameworks that create sequences synthetically and are unable to model them on a biologically relevant data set.

While this implementation uses a base-set of sequences (such as 3′UTR and 5′UTR transcripts) to generate datasets, future work involves using a generative model such a $n$-gram model to generate sequences (See Chapter 3 Section 3.5). Generative models learn the distribution of the 3′UTR and 5′ UTR sequences, and then can be subsequently used to generate sequences that simulate 3′UTR and 5′ UTR transcripts. Benchmarking the performance of computational methods when both these methods used to generate sequences is a candidate for future work. The next section discussed strategies that could be employed to validate the generated sequences.

“Truth” file: In order to document the signal embedded as well as the location of the embedded signal to the positive set of sequences, the PySG framework creates a file called the “Truth file”. This file contains the sequence id (gene id, transcript id, gene name), the signal embedded, and the location at which the signal is embedded. This file comes into play when evaluating the predicted motifs by the computational tools. An example of this file is as shown below:

```
ENSG00000102312|ENST00000355961|PORCN,GAGAAA,10
ENSG00000080189|ENST00000372230|SLC35C2,AAACGT,25
ENSG00000213465|ENST00000246747|ARL2,CCCGTT,35
```
Figure 5.10: Comparison of 3-grams of 3’UTR and 5’UTR transcripts and created positive and negative sequences.

5.3.4 Step 4: Running Computational tools

To evaluate the performance of computational tools, the generated sequences from Section 5.3.3 are run through the computational tools registered with the PySG framework. Currently, the PySG framework only supports two discriminative tools: DREME [13] and the k-spectrum kernel method [164]. The choice of these tools was motivated by the results of the techniques in predicting binding sites for RBPs HuR and TTP. (See Chapter 4). The first goal of developing the framework was to be able to test-drive it end to end and hence only two computational tools
were added. Augmenting the framework to support either a different type of signal, or a new computational method is not very hard, and Section 5.4 outlines the effort. Default parameters for DREME were used; whereas for the $k$-spectrum kernel method, 10-fold cross validation is used and the $k$-mer lengths were set between 6 and 15 to match the default configuration of DREME [13]. Each computational tool was run sequentially on a set of positive and negative sequences under the experimental conditions set up by the configuration file.

The output of DREME is a text file that contains the position weight matrix (PWM) (See Appendix B) of the top motifs identified in the sequences, whereas the $k$-spectrum kernel outputs the $k$-mers that were used to discriminate between the positive and negative set of sequences. These output files are parsed and compared in the next step.

5.3.5 Step 5: Evaluate Discriminative methods

In order to assess the predicted signal, and compare them to the embedded signal, the result files generated in Step 4 were first parsed. The parsers are a part of the PySG framework and written in Python.

![Figure 5.11: Overview of evaluating motifs identified by DREME](image)

**DREME evaluation:** The motifs with high E-values identified by DREME [13] were parsed from “dreme.txt” file for the experiment run. DREME [13] provides a PFM associated with each motif. The PFMs for the experiment run are parsed and stored as a list of PFMs. Each PFM from the list is input to the TAMO [68] MotifTools library. This library has an API called `bestscanseq` which when input a PFM, and a sequence, outputs the best $k$-mers for the provided PFM file.
The best $k$-mers that a motif could match with a sum (log-offs) score that is greater than the threshold. By default, the threshold value is $75\%$ of the maximum score [68]. The threshold minimizes false positives (how many shuffled or random sequences does the PSSM match with that score) and false negatives (how many of the “real” sequences are missed with that score). If the threshold selected is too stringent, only the strongest true-positive motifs will be identified, and the weaker ones will be ignored. Next, a regular expression is created which is a union of all the $k$-mers. The regular expression is then searched for in each sequence to find predicted matches. The output of the search is saved as a key, value pair where the key is a combination of sequence id, start location, and end location and the signal identified is compared to the “Truth” file (Section 5.3.3). This is carried out for every sequence. Figure 5.11 illustrates provides a flow chart of the evaluation process for DREME [13] and the evaluation of the predicted and embedded motifs is illustrated using Figure 5.12 and 5.13.

$k$-spectrum evaluation: The output of the $k$-spectrum kernel method is a list of discriminative $k$-mers that are scored and the score identifies the significance of the $k$-mer. The initial approach was to create a PFM using top-$N$ $k$-mers, and follow the same process as carried out with DREME. However, the $k$-spectrum kernel method provides definite $k$-mers that preserve dependencies between motif positions, unlike PSSMs. PSSMs also have a tendency to underfit data and thus are unable to capture discriminative sequence features. Instead, a regular expression was created by selecting the top-$N$ $k$-mers. The regular expression is matched for every sequence, and the location of the predicted $k$-mer(s) was compared to the location of embedded signal.

The computation of sensitivity and PPV is done on a nucleotide basis and is illustrated using Figure 5.12 and 5.13. Figure 5.12 illustrates the location of embedded signal and discovered signal. In this example, the discovered signal is a few nucleotides off from the embedded signal. The nucleotides between the start of the predicted signal and embedded signal are considered as “false negatives”, and the $k$-mer nucleotides that match the embedded $k$-mer are considered as “true positives”, the nucleotides of the predicted $k$-mer that are beyond the end of the embedded signal are considered to be “false positives”.
Figure 5.12: The top sequence shows the embedded k-mer and the bottom sequence shows the predicted k-mer. The starting position of the predicted k-mer is greater than the start position of the embedded motif. Hence, $falseNegatives = predictedStart - embeddedStart$. Next, the embedded k-mer nucleotides are compared with that of the predicted k-mer. In this example, the predicted k-mer matches until the end of the embedded k-mer hence $truePositives = embeddedEnd - predictedStart$, and finally, $falsePositives = predictedEnd - embeddedEnd$.

Figure 5.13: In this case, the predicted k-mer is found after the end of the embedded k-mer. Thus, the entire length of the embedded k-mer is the number of false negatives or $falseNegatives = embeddedEnd - predictedEnd$. $falsePositives = predictedEnd - predictedStart$. $truePositives = 0$ as the computational tool did not find any motifs at the location where the signal was embedded.

Figure 5.13 illustrates the case where the predicted signal is outside the boundaries of the embedded signal. In this case, the nucleotides that fall between the coordinates of embedded signal (denoted by the dotted lines) are considered to be “false negatives”, and the nucleotides of the predicted k-mer are considered to be “false positive”. This is carried out for every sequence in both positive and negative sequences and the sensitivity and PPV values are computed.
Alternative approach: While the approach discussed evaluated the presence of a motif at a given location, an alternative method to consider is to evaluate the method on its ability to identify the embedded motif in the sequence. The sequence-based approach considers a sequence as a binder if any of the predicted motifs contain embedded signal. If the predicted motifs do not contain the embedded signal, the sequence is considered to be a non-binder and the false negative count is increment by one. However, when the predicted motif is found by the computational method, the sequence is considered to be a binder, and the number of true positives is incremented by one. The negative set should not contain any signals embedded in them. If the motif is found in the negative set, the number It is important to note that the base expectation of finding a motif in any sequence is random (50%) whereas the probability of finding a motif at a specific location in a sequence is very close to zero. The nucleotide-based approach would be ideal when dissecting the precise binding site for further analysis (such as mutations) is important. On the other hand, for a given set of sequences, the presence or absence of binding sites needs to be evaluated, the sequence-based approach could be used.

5.3.6 Step 6: Benchmark results

To benchmark the performance of computational tools DREME and the $k$-spectrum kernel method, each experiment is run five times, and each run, the sensitivity, and PPV are computed. This is carried out to ensure the values of sensitivity, and PPV is not specific to a particular set of generated sequences. Each set of sequences is generated from random locations from the 3′UTR and 5′UTR sequences. The mean and standard error is computed for both sensitivity and PPV (Step 5), and error bars are plotted for DREME [13] and the $k$-spectrum kernel method. The $k$-spectrum kernel method has three different sensitivity and PPV values based on the number of discriminative k-mers used when evaluating the results.
5.4 Augmenting the framework

The goal of this project was to be able to architect a framework that would automate the generation of sequences that contained a user-defined signal and evaluate the performance of computational tools. The first step was to verify the functionality of such a framework, and secondly, build it in such a way that augmenting it (either adding new signal types or new computational tools) should be fairly trivial. This section will describe these modifications.

```python
if __name__ == "__main__":
    import sys
    confFile = sys.argv[1];
    uuid_to_append = str(uuid.uuid4())
    generator = YamlFastaGenerator(confFile, uuid_to_append);
    targetDir = generator.GetTargetDir();
    generator.CreateConfFiles();
    GenerateFastaFile(targetDir);
    RunComputationalTools(targetDir);
    ParseResultsAndGenerateGraphs(targetDir)
```

**Figure 5.14:** The figure illustrates the simplicity of the framework and the lines of code map to the Sequence Generator architecture in Figure 5.8. Adding a new signal type will require changes to the function `CreateConfFiles()` which generates the individual experimental configuration files and the function `GenerateFastaFile()` which will embed the new signal type in the FASTA files. Adding a new computational method to the framework will require changes to functions `RunComputationalTools()` and `ParseResultsAndGenerateGraphs()`. See the next figure for more details.

```python
def RunComputationalTools(directory):
    os.chdir(directory);
    for signalFile in findFiles(directory, "Signal\.*.fa"):
        signalFileName = os.path.basename(signalFile);
        noSignalFile = dirName + "/NoSignal\.*.fa"
        dremeResultDir, realKmersCsvFile = RunComputationalMethods.RunDremeAndGetResults(signalFile, noSignalFile);
        kspectrumResultDir, realKmersCsvFile = RunComputationalMethods.RunKspectrumAndGetResults(signalFile, noSignalFile);
```

**Figure 5.15:** The figure illustrates the `RunComputationalTools` method that iterates through the directory looking for Signal and NoSignal files, and sequentially runs through the computational tools. DREME and k-spectrum are examples shown. This function will require an addition of a function call to add a new computational method to the framework. The individual method will then invoke the computational tool passing in the right parameters.

**Adding new types of signals:** Circling back to Figure 5.8, adding a new signal involves changing three steps of the framework: Step 1 and Step 3. This requires modifying the Python code to handle a new type of signal (Step 1) which will generate individual experiment files that
contain the new signal type. Step 3 will need to parse the new signal type, and based on the signal type, generate the motif to embed in the FASTA sequences of the positive set. See Figure 5.14

**Adding new computational tool:** Referring back to Figure 5.8, the addition of a new computational tool requires slightly more effort when compared to adding a new signal. However, the effort is confined to Step 4 where computational tools are run sequentially (adding the line to run the computational tool) and Step 5 where the results of the new computational method need to be parsed so that they can be compared to the embedded signals. Step 5 will require writing a parser to dissect the predicted motif from the output of the computational tool followed by saving it to a file where the comparison takes place. The comparison module is written in such a way that if the files that contain the embedded signal (“Truth file”) and predicted signal are done correctly, the framework should work seamlessly. See Figure 5.14 and 5.15.

This section described the ease of augmenting the framework and outlined the methods that need to be modified to add new signal types or new computational methods. The next section will provide guidelines to consider when using the sequence simulator component of the framework to generate sequences.

### 5.5 Guidelines for generating sequences using PySG framework

Section 5.2 outlined the biochemical mechanisms that are involved when RBP binds to mRNA target. Understanding the binding preferences RBPs exhibit is paramount before using the sequence simulator module of the framework to generate sequences.

**Binding location:** For a given RBP, the binding location within the target RNA is typically consistent e.g. 3’ UTRs, introns, coding regions. For example, PTB is an RBP that is known to bind to the 3’ end of introns and the binding site is pyrimidine-rich in nature whereas HuR and TTP generally bind to the 3’ UTR regions. When generating sequences, the sequence simulator allows users to select the background set of sequences that will be used to generate the positive datasets. As seen in Step 3, the base-set of sequences used to generate positive set were 3’ UTR and 5’ UTR transcripts to mimic the binding preferences of HuR and TTP. However, there are
RBPs that bind to introns, or coding regions. This would require the base set of sequences to be modified appropriately. Such specific binding characteristics of RBPs can be modeled by carefully developing the base-set of sequences. Additionally, if the targets of the RBP being studied are known to be well-conserved, the base-set could include sequences from multiple organisms.

**Figure 5.16:** Components involved in the binding of RBPs to target mRNA

**Binding domain:** After the target regions are handled correctly, understanding the biochemistry of the binding domains is important before sequences can be generated. Based on the RBP involved in the binding, the domains involved in the binding could be 50-100 nucleotides in length. These RNA binding domains (RBDs) are well studied, grouped into families/clan based on sequence specificity, and their sequences are available in the Pfam database [56]. Figure 5.16 depicts the different components involved in the RBP binding to a target RNA. While the RBD spans over a few nucleotides, the contact point with the RNA is typically shorter in length (e.g. 4-8 nucleotides in the case of RRM). The contact points can be represented as motifs and the position weight matrix associated with the motif can be obtained from CIS-BP-RNA [150] database. The framework currently supports different representations (position weight matrix, position frequency matrix, secondary structure alignment) of embedding the motif into the sequence. However, the RBDs are not integrated into the framework when creating the positive dataset. However, given the modular nature of the PySG framework, integrating the sequences from the Pfam database [56] would require changes outlined in Figure 5.17.

The first and the third step are already handled in the current architecture. However, the
Figure 5.17: Generation of positive sequences: Improved process to include RNA binding domain followed by embedding the motif

The framework needs to be modified to incorporate Step 2 and this can be achieved by doing the following:

1. Update the experiment configuration file (outlined in Section 5.3.1) to add rbdSequence-File. This file will contain RBD-specific sequences obtained from Pfam database for the RBP that is being studied.

2. Randomly select sequences from the RBD sequence file and add them to the target RNA fragment. (outline in Section 5.3.3)

3. Insert the signal associated with the RBP within the RBD. Based on the number of contact points made with the RNA, the number of motifs would vary.

Negative Set: Currently, the negative set is created by shuffling the sequences in the positive set. While the positive dataset is focused on simulating the binding preferences of the RBP involved, the negative set could be generated from a separate pool of sequences that are not involved in the
binding of the studied RBP.

5.6 Discussion and Future Work

This chapter described the architecture of the PySG framework to generate sequences followed by benchmarking them under various experimental considerations. While most sequence simulators focus on embedding mutations or sequencing errors, this framework solves the problem of evaluating different computational techniques based on simulating various biologically relevant conditions. The developed framework is modular, scalable to include new signal types and additional computational methods, easy to use as it involves one experimental driver configuration file, and first of its kind to benchmark computational methods to identify RBP targets. The novelty of this architecture coupled with the easy-of-use makes this framework a valuable resource to the bioinformatics community. There are several features that could be added to the framework and the list captures them:

- Adding support for a Hidden Markov Model (HMM) is another way to enrich the list of signals supported by this framework.

- Augment the framework with structure prediction methods (See Chapter 6).

- To ensure the RBP characteristics are considered during dataset creation, it is important to have RNA binding domain (RBD)-specific sequence in addition to the contact point (motif). Necessary details for this item are outlined in Section 5.5.

- TAMO uses a cut-off to identify best motifs for a sequence given a PSSM associated with DREME motif. It is possible to vary that cut-off and evaluate the effect by plotting a ROC curve that examines the changes to the false positive and true positive rates as threshold value changes.

- Allow multiple instances of a particular signal to be present in the same sequence.

- Allow multiple types of signal to be embedded in the same sequence.
• Override the default values of computational methods. Currently, regardless of the motif type or length, the default motif size is used for both DREME and the $k$-spectrum kernel method.

• Create background sequences using a generative language model.

• Validation of generated sequences. See Section 5.6.1

• Compare the performance of computational methods when the background sequences are generated using an $n$-gram model and random fragments from the 3′UTR and 5′UTR sequences.

• Randomly select $2N$ number of sequences where $N$ sequences are used for the positive dataset, and the other $N$ are used for the negative dataset. While the trinucleotide distribution will not be consistent across both the datasets, but it would eliminate the need to shuffle to positive sequences in order to generate the negative set of sequences.

5.6.1 Validating sequences generated using a model

Section 5.2 details the binding preferences of RBPs which characterizes the nature of sequences that should be generated when a model is used to create synthetic datasets. In the rest of this section, the term generated set will be used to refer to the set of sequences generated using a model and experimental set to refer to the set of sequences generated using an experimental method (examples of experimental methods for RBPs are discussed in Chapter 2, Section 2.3).

Nucleotide distribution: Generate $N$ sets of generated sequences and compute the di/trinucleotide composition for each set. The average and variance of the nucleotide distribution should be plotted and then compared to that of the held-out set (experimental set). A close match of the distribution ensures that the generated set of sequences do indeed resemble the experimental set.

Motif searching: Run $N$ sets of generated sequences through a motif searching tool such as MEME [12] to identify the over-represented motif ($M_{generated}$). Run MEME on the experimental set to identify the highly represented motif ($M_{experimental}$). Using the API bestseqs from
the MotifTools library in TAMO [68], the sequences in the experimental set that contain the \(M_g\)enerated motif can be identified and vice-versa. The validity of the generated dataset would be confirmed if the score of the \(M_g\)enerated motif is high in the experimental set and the score of the \(M_e\)xperimental motif is high in the generated set.

**Calculation of Percent Identity:** Perform multiple sequence alignment of the sequences in the generated set and the sequences in the experimental set. Percent identity is computed by multiplying the number of matches in the pair of sequences by hundred and dividing this by the length of the aligned region which includes gaps. Species that are closely related have higher percent identity than those that are distantly related ones. Thus, higher percent identity would ensure the validity of the generated set of sequences.

**Discriminative Method:** Create a positive set comprising of generated set of sequences and a negative set comprising of the experimental set of sequences. If a discriminative method such as DREME [13] is unable to identify an over-represented motif would indicate the validity of the generated set.

The next chapter will focus on using the framework to benchmark the \(k\)-spectrum kernel method, and DREME [13] by varying the conditions such as signal-to-noise ratio, and the percentage of sequences containing the signal, as well as embedding different types of motifs (primary sequence, and secondary structure).
Chapter 6

Benchmarking Results

Chapter 5 described the overall architecture of our framework for benchmarking RBP motif discovery tools. The goal of this chapter is to provide a basic framework in order to both test RBP inference methods and test drive the framework. In order to do so, two biological conditions will be simulated using the PySG framework: (i) changing signal-to-noise ratio: where the length of the sequences generated is varied (noise is increased), and (ii) number of sequences in the positive set that contains embedded signal is varied thus varying the signal percentage of the sequences. The signal embedded in these sequences is discussed in Section 6.1. Section 6.2 outlines the setup for each experiment, followed by detailing the experiment and analysis of the results in Section 6.3 and 6.4. The chapter concludes by discussing the future work in Section 6.5.

6.1 Selection of signals: primary and secondary structure

As described in Chapter 2, RBPs recognize either primary sequence or secondary structure for their binding targets. Biologically, the primary sequence motifs have two different variables: a) information content and b) length of the motif. The information content refers to the variability that exists in nucleotides of the motif. The length refers to the number of nucleotides present in the motif. Using the above-mentioned criteria, four primary motifs were selected and are illustrated in Figure 6.1. The HuR and TTP motifs are high in information content and this is evident with 3-4 nucleotides having a large size and there exist some variations in the remaining 5-6 nucleotides. These are both 9 nucleotides long. The FUS motif is a short but specific motif whereas
the LIN28 motif is around 7 nucleotides long but has lower information content. The selected four RBPs have variability in length and information content. In addition, the motifs of the selected RBPs have different nucleotide composition thus preventing any bias towards particular nucleotide distribution.

![Sequence logos of sites bound by RNA binding proteins that recognize primary sequence. These four RBPs were benchmarked, and their motifs were obtained from CIS-BP-RNA](image)

**Figure 6.1:** Sequence logos of sites bound by RNA binding proteins that recognize primary sequence. These four RBPs were benchmarked, and their motifs were obtained from CIS-BP-RNA [150]

**Secondary structure signal:** To embed secondary structure signal, existing RNA structural elements sequences published in the Rfam database [138] were used. Rfam DB [138] is a widely used database and it contains a collection of multiple sequence alignments, consensus secondary structures, and covariance models for RNA families. In addition, the database contains the multiple sequence alignments (MSA) for RNA structural elements and are represented in FASTA format. Three simple structures known to contain stem-loop/hairpins were selected from this database. The selected structures were Iron Response Element (IRE), LIN-4, and micro RNA mir-101 (See Figure 6.2 for structure representations).

Each RBP and RNA structural element were benchmarked, however here I present benchmark results for only one RBP and one structure element. Appendix E contains the results for the other
Figure 6.2: Sequence logos of RNA structure elements. Figures for each structure is obtained from Rfam DB [138]
RBPs and structure elements.

6.2 Experimental Setup

**General setup:** The setup for every experiment consists of creating sequences with an embedded signal, followed by benchmarking the performance of computational methods on the generated set of sequences. To remove any preference for a specific set of created sequences, each experiment configuration being tested was run five times. Creation of sequences is discussed in detail in Chapter 5, Section 5.3.3. For all experiments, the number of sequences was kept constant at 512. It is important to note that this number represents the number of sequences created per set; 512 positive and 512 negative sequences were created. Sensitivity and PPV were computed for each run and the mean and standard error were represented as error bar plots. The length of the motifs identified was set to default values which for DREME is 6-15 nucleotides. The $k$-spectrum $k_1$ and $k_2$ were matched to those values. By using default values, one gets the out-of-the-box results from a computational method. However, careful tuning of parameters of each computational method based on the dataset would result in better performance. However, the ability to set parameters for each computational tool is not currently supported by the PySG framework.

**Sensitivity/PPV:** Sensitivity also known as true positive rate, is computed as $\frac{TPs}{TPs + FNs}$ whereas **Positive Predictive Value** (PPV) is evaluated as $\frac{TPs}{TPs + FPs}$, where $TP$ is the number of true positives, $FN$ is the number of false negatives, and $FP$ is the number of false positives. Sensitivity measures the proportion of positives that were correctly identified. In this case, the proportion of the embedded motifs that were identified by the computational tools correctly. PPV measures the proportion of predicted positives that are correctly identified.

Next section details the individual experiments run and discusses the findings of each experiment.
6.3 Evaluate the effect of varying “signal-to-noise” ratio

In order to evaluate the effect of varying “signal-to-noise” ratio, in this experiment, the signal was embedded in varying length of sequences (varying noise levels).

**Biological significance of this test:** RBPs recognize binding sites along the mRNA. The number of RBPs bound to the mRNA affects the regulation of the resulting message [182]. Pancaldi et al. [141] further found that highly translated RBPs are known to bind to shorter mRNAs. Thus indicating that the length of the transcript does play an important role in RBP binding.

6.3.1 Primary Sequence motifs

**Objective:** Evaluate the performance of computational methods when the signal is a primary sequence, and “signal-to-noise” is varied.

**Test Setup:**

Listing 6.1: Experiment Configuration for RBP LIN-28

```python
targetDir: 'Lin28_Test_SeqLength_Shuffle'
nomSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
    type: 'pfmFiles'
pfmFiles: ['LIN28A_1.pfm']
signalPercent: [100]
osignal:
    type: 'shuffle'
fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```

Listing 6.1 illustrates the experiment configuration for RBP LIN-28. The “signal-to-noise” ratio was tested by varying the sequence length keeping the embedded motif the same. Given that the RBP experimental methods provide sequences of 100 – 200 in length but may only include a portion of recognized motif, we test lengths 128 – 1024 nucleotides. The LIN-28 motif was obtained from CIS-BP-RNA [150] and was embedded in all sequences of the positive set. In order to keep the experiment focused on sequence length variation, all other parameters were kept constant. Table 6.1 enumerates the average GC-content of sequences for each experiment carried out. Each
experiment is run five times, and the variance depicts the variation in the GC-content across the runs.

<table>
<thead>
<tr>
<th>Sequence Length</th>
<th>Mean GC-Content%</th>
<th>Variance GC-Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>48.84</td>
<td>0.13</td>
</tr>
<tr>
<td>256</td>
<td>47.15</td>
<td>0.15</td>
</tr>
<tr>
<td>512</td>
<td>45.11</td>
<td>0.18</td>
</tr>
<tr>
<td>1024</td>
<td>43.71</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Table 6.1: The mean and variance of the GC-content for the generated sequences for each value of sequence length. Five sets of sequences are generated for each value of sequence length, both mean and variance are computed.

**Observation and Interpretation:** The top \( k \)-mers identified by the \( k \)-spectrum kernel method are unable to capture the signal, this is indicated in the Figure 6.3 by lower values of sensitivity associated with the top-25 (green line much below the yellow line). With the increase in the number of discriminative \( k \)-mers, the sensitivity value increased by 3-4 percentage points. Secondly, in general, the \( k \)-spectrum had higher values of sensitivity when compared to DREME. Both the methods perform better with less noise (shorter sequences), but DREME has higher PPV values at lower noise. This suggests that the \( k \)-spectrum kernel method has a higher false positive rate that is rather invariant to the sequence length.

Qualitatively, the LIN-28 motif logo is primarily GA-rich and this trend is seen in the top-25 \( k \)-mers when the sequence length is 128 nucleotides (See Appendix F). The motifs identified by DREME as well as \( k \)-mers by the \( k \)-spectrum kernel method, however, start picking up background noise as the sequence length is increased and this is evident with the drop in sensitivity values. Appendix F lists the \( k \)-mers and the motifs identified by both the computational methods. As the noise levels increased (sequences got longer), the average GC-percentage of the sequences dropped from 48% to 43%. Future work could address generating datasets that maintain GC-percentage.

### 6.3.2 Secondary structure motifs

**Objective:** Evaluate the performance of primary sequence discriminative methods when the signal is secondary structure, and “signal-to-noise” is varied.
Figure 6.3: Primary Sequence Motif: RBP LIN-28: Effect of varying the signal-to-noise ratio. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

Test Setup:

Listing 6.2: Experiment Configuration for RBP LIN-4

```plaintext
targetDir: "RF00052_Test_SeqLength_Shuffle"
numSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
  type: "structure"
  structureFiles: ["RF00052.fa"]
  signalPercent: [100]
global:
  type: "shuffle"
  fastaFile: "3UTR_5UTR_Transcripts_Human.fa"
```

Listing 6.2 illustrates the experiment configuration for RNA structure element LIN-4 microRNA precursor. As with the previous experiment, the number of sequences was kept constant at 512 and the sequence length was varied similarly from 128 to 1024 nucleotides. The signal in this experiment are the multiple sequence alignments for the RNA structure element LIN-4 microRNA precursor. These alignments are saved in FASTA format in the file RF00052.fa. An example of
sequence alignments is listed in the Listing 6.3. The conserved and co-varying regions are separated by spaces. These are fairly long sequences compared to the primary sequence recognition motifs. However, the defaults used by either of the computational tools were not modified (6 – 15 motif size). The hypothesis was that both computational methods would identify sub-sequences (sub-strings) of the longer embedded structure motif.

**Listing 6.3:** Multiple sequence alignments associated with LIN-4

```
CTGTCT CTTGT TTCAAC AGTGTTTG GACAGAACAG
TGGGTT CTTGC TTCAAC AGTGTTTG AACGGGACCCG
CTGTCT CCGGC TTCAAC AGTGTTTG GACGGAACAG
CCGCGT TCTCG CTTCAA CAGTGCTT GA A C G G A A C GG
GAGGTT CTTGC TTCAAC AGTGATTG AACGGAACTTC
```

Table 6.2 enumerates the average GC-content of sequences for each experiment carried out. Each experiment is run five times, and the variance depicts the variation in the GC-content across the runs.

<table>
<thead>
<tr>
<th>Sequence Length</th>
<th>Mean GC-Content%</th>
<th>Variance GC-Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>50.24</td>
<td>0.56</td>
</tr>
<tr>
<td>256</td>
<td>48.11</td>
<td>0.11</td>
</tr>
<tr>
<td>512</td>
<td>45.78</td>
<td>0.19</td>
</tr>
<tr>
<td>1024</td>
<td>43.93</td>
<td>0.05</td>
</tr>
</tbody>
</table>

**Table 6.2:** The mean and variance of the GC-content for the generated sequences for each value of sequence length. Five sets of sequences are generated for each value of sequence length, both mean and variance are computed.

**Observation and Interpretation:** As seen in Figure 6.4, the sensitivity and PPV dropped with the increase in noise levels for DREME but did not have any impact for the $k$-spectrum kernel method. However, in this experiment, DREME, a primary sequence motif identification method, out-performed the $k$-spectrum kernel method. DREME identifies motifs as compared to the $k$-spectrum kernel that identifies $k$-mers. Motifs account for the variations that are typical when secondary structures are involved and implicitly capture the covariation. PWMs are known to capture secondary structures better than fixed $k$-mers. The number of $k$-mers made a very little difference to the sensitivity and PPV values (less than one percentage points) as they did when the
motif was primary sequence-based. The sensitivity values were within one percentage point of each other (top-25, top-50, and top-100). As the noise levels increased, PPV of DREME was closer to that of the $k$-spectrum kernel method.

![Graph showing the effect of sequence length on sensitivity and PPV](image)

**Figure 6.4:** Secondary Structure Motif: RNA structure element: LIN-4: Effect of varying the signal-to-noise ratio. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

The next section will test the effect of varying the number of sequences that contain the embedded signal for both types of motifs.

### 6.4 Evaluate the effect of varying “signal-percentage”

Signal-percentage refers to the number of sequences in a dataset that contains the embedded signal. In order to evaluate the effect of varying “signal-percentage”, in this experiment, the number of sequences that contained the signal was varied. The same RBP LIN-28 (primary sequence) and RNA structure element lin-4 microRNA precursor (secondary structure) were the signals embedded and were run on both computational methods (DREME and $k$-spectrum kernel method).
6.4.1 Primary Sequence motif

**Objective:** Evaluate the performance of computational methods when the signal is a primary sequence, and “signal-percentage” is varied.

**Biological significance of this test:** Advances in experimental methods have led to single nucleotide resolution when identifying RBP targets. However, experimental methods as discussed in Chapter 2 Section 2.4 have shortcomings, and one of them being the fact that there might be reads that do not contain the binding site due to errors in sequencing. Thus, this experiment attempts to simulate this scenario: not all sequences in a dataset contain the signal (binding site).

**Test Setup:**

Listing 6.4: Experiment Configuration file for RBP LIN-28

```yaml
targetDir: 'Lin28_Test_SignalPercent_Shuffle'
numSeq: [512]
seqLen: [256]
signal:
  type: 'pfmFiles'
pfmFiles: ['LIN28A_1.pfm']
signalPercent: [10, 25, 50, 75, 100]
noSignal:
  type: 'shuffle'
fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```

Listing 6.2 illustrates the experiment configuration for RNA structure element lin-4 microRNA precursor. As with the previous experiment, the number of sequences was kept constant at 512, and the sequence length was kept constant at 256 nucleotides. From the previous Section 6.3.1, it was noted that the sequence length greater than 256 nucleotides impacted the quality of the motifs identified by the computational methods. The signal percentage (percentage of sequences that contained the embedded signal) was varied between values [10, 25, 50, 75, 100] and the sensitivity and PPV values were plotted. The LIN-28 motif was obtained from CIS-BP-RNA [150] and was embedded in all sequences of the positive set. Listing 6.4 illustrates the configuration used for the experiment. The hypothesis for this experiment was that as the signal percentage increases, the sensitivity would increase linearly and probably flatten after a certain percentage.
<table>
<thead>
<tr>
<th>Signal %</th>
<th>Mean GC-Content%</th>
<th>Variance GC-Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>47.35</td>
<td>0.04</td>
</tr>
<tr>
<td>25</td>
<td>47.48</td>
<td>0.15</td>
</tr>
<tr>
<td>50</td>
<td>47.42</td>
<td>0.09</td>
</tr>
<tr>
<td>75</td>
<td>47.55</td>
<td>0.42</td>
</tr>
<tr>
<td>100</td>
<td>47.86</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 6.3: The mean and variance of the GC-content for the generated sequences for each value of sequence length. Five sets of sequences are generated for each value of sequence length, both mean and variance are computed.

**Observation and Interpretation:** In general, the $k$-spectrum had higher sensitivity than the DREME computational method. The sensitivity values of the $k$-spectrum kernel improved slightly when 50% or more sequences contained the embedded signal possibly suggesting that this method would be a good candidate when noisy data is involved. When only 10% of the sequences contained the signal which translates to very noisy data, the $k$-spectrum did surprisingly well as compared to DREME. While DREME had higher PPV values as the signal percentage increased, at 10%, the $k$-spectrum kernel had higher sensitivity and PPV values thus confirming the use of this method in case of noisy data. Once again, DREME had better PPV and it increased as more sequences contained the signal. The PPV values of DREME had higher variation than those for the $k$-spectrum kernel method. When all the sequences contained the signal, the sensitivity of both the methods was comparable.

To summarize, in this experiment, it was evident that the $k$-spectrum kernel was able to capture the embedded signal when only a fourth of the sequences contained the signal. DREME [13] saw a sharp reduction in false positives when 50% of the sequences contained the signal, thus indicating that the $k$-spectrum kernel is a better choice when the number of sequences containing the signal is low. The next section will evaluate the effect of signal percentage when the motif is secondary structure.
6.4.1.1 Secondary structure Recognition Motifs

Objective: Evaluate the performance of primary sequence discriminative methods when the signal is secondary structure, and “signal-percentage” is varied.

Test Setup: Similar to the previous test in Section 6.4.1, the sequence length was kept constant at 256 nucleotides. The signal in this experiment are the multiple sequence alignments for the RNA structure element LIN-4 microRNA precursor. These alignments are saved in FASTA format in the file RF00052.fa. An example of multiple sequence alignment embedded into the positive sequences is shown in Listing 6.6. The conserved and co-varying regions are shown in red color. These are fairly long sequences compared to the primary sequence recognition motifs. However, the defaults used by either of the computational tools were not modified (6−15 motif size). The hypothesis was that both computational methods would identify sub-sequences (sub-strings)
of the longer embedded structure motif.

Listing 6.5: Experiment Configuration for structure element LIN-4

```json
targetDir: 'RF00052_Test_SignalPercent_Shuffle'
umSeq: [512]
seqLen: [256]
signal:
  type: 'structure'
  structureFiles: ['RF00052.fa']
signalPercent: [10, 25, 50, 75, 100]
global:
  type: 'shuffle'
  fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```

Listing 6.6: Multiple sequence alignment motifs embedded for lin-4 microRNA precursor

```
CTGTCT CTTGCTTCA AC AGTGTTTG AACAGAACAG
CTGTCC CTTGCTTCA AC AGTGTTTG GACAGAACAG
AGAGTT CTTGCTTCA CA GTGTTTGA ACGGAACCCT
CGGTTT CTTGCTTCA AC AGTGCTTG GA C G G AA G C C G
GAGGTT CTTGCTTCA GC AGTGTTTG GACGGAACCTC
GAGGTT CTTGCTTCA GC AGTGTTTG GACGGAACCTC
```

<table>
<thead>
<tr>
<th>Signal %</th>
<th>Mean GC-Content%</th>
<th>Variance GC-Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>47.14</td>
<td>0.43</td>
</tr>
<tr>
<td>25</td>
<td>47.62</td>
<td>0.13</td>
</tr>
<tr>
<td>50</td>
<td>47.97</td>
<td>0.04</td>
</tr>
<tr>
<td>75</td>
<td>48.24</td>
<td>0.32</td>
</tr>
<tr>
<td>100</td>
<td>48.00</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Table 6.4: The mean and variance of the GC-content for the generated sequences for each value of sequence length. Five sets of sequences are generated for each value of sequence length, both mean and variance are computed.

**Observation and Interpretation:** As seen in Figure 6.6, DREME [13] out-performed the k-spectrum kernel method in case of sensitivity and PPV values. The step-wise increase in the signal percentage level, showed an increase in sensitivity for DREME, however, after 50% of the sequences containing the signal, the increase in sensitivity was not as drastic. This indicates that DREME was able to identify the embedded signal with 50% of the sequences containing the embedded signal. However, the number of false positives decreased with the increase in signal percentage for DREME. This can be seen with a linear increase in PPV values with every step increase in signal percentage.
The sensitivity values of the $k$-spectrum kernel method matched those for DREME when 10% of the sequences contained the embedded signal, but the kernel method was unable to keep up after that point. Since the $k$-spectrum kernel method had higher sensitivity values when the motif was primary sequence based and at low signal percentages but does poorly when the motif is secondary structure based, it is evident that the $k$-mers identified once again did not capture the covariation that existed in the embedded signals.

![Graph](image)

**Figure 6.6:** Secondary Structure Motif: RNA structure element: LIN-4: Effect of varying the signal- percentage. The number of sequences that contain the embedded signal is varied between 10 percent and 100 percent. The sensitivity and PPV values are computed at every level of signal percentage.

The four experiments test drove the PySG framework and demonstrated its ability to: (i) simulate biologically relevant conditions, (ii) benchmark computational methods, (iii) modify the motif that is embedded in the generated sequences, and (iv) run to completion i.e. from configuration to graphs as outlined in Chapter 5. The next section will summarize the findings of the
benchmarking results.

6.5 Discussion

This chapter demonstrated the use of the PySG framework to both simulate sequences as well as benchmark computational methods. The simulated scenarios described in this chapter were examples of using the framework but not necessarily the exhaustive list of experiments that could be run using the framework.

By benchmarking both primary sequences as well as secondary structure signals under two biological conditions, it was evident that the $k$-spectrum kernel method was able to identify primary sequence based signals. Using the top-100 kmers, it was able to capture the signal even when the percentage of sequences that contained the motif was as low as ten percent. This makes the $k$-spectrum kernel method an ideal candidate when the motif is primary sequence based and not all sequences contain the motif (could be due to experimental method limitations). However, one drawback to using the $k$-spectrum kernel method was a high number of false positives which was evident with low PPV values on the graphs.

Neither the $k$-spectrum kernel method nor DREME [13] are designed to identify secondary structure signals, however, DREME did surprisingly well at identifying the covarying regions of the embedded secondary structure signals. This was due to the wildcard approach that is used by DREME in identifying motifs that are useful when there is covariation present in the motifs. DREME, when used to identify secondary structure signals, did hit that cut-off value when the sequences were 256 nucleotides long. However, the low sensitivity values can be attributed to the stringent computation of what constitutes a true positive, a false positive, and false negative. The framework used nucleotide-based comparison where only the region under consideration was the location of the embedded signal. Any signal found outside the range of the embedded signal incremented the number of false positives. For example, consider a sub-sequence such as AU-AUAUAUAUA from locations 18 – 27, and the embedded k-mer the first AUUA is from nucleotides 18 – 22, but the computational tool identifies the last AUUA from location 24 – 27, it would be
considered to be a false positive even though it actually identified the right motif.

One other aspect of the discussion was the drop in performance when the sequence length is increased. As the sequence length increased i.e. the noise level increased, the performance of both the computational methods dropped. This is an artifact of signal-to-noise ratio (SNR) dropping. Let us consider an example where the sequence length is set to 128 nucleotides and the motif is 8 nucleotides long. The corresponding SNR, in this case, is 1/16 or 0.0625. As the sequence length is increased to 256 but the motif length stays the same, the ratio drops further to 0.0325 and as the sequence length is increased to 1024 nucleotides, the ratio is down to 0.007812. Thus, the noise is much much greater than the signal which results in the computational methods identifying motifs in the noise. As the length increases, a good next test to run would be to increase the number of motifs to keep the original SNR constant and re-evaluate the performance of the computational methods.

When the signal was primary sequence-based, the $k$-spectrum kernel method had higher sensitivity values but low PPV values. This suggests that the kernel method had a higher number of false positives. Sensitivity and specificity are often trade-offs made when using discriminative methods. When computational methods such as the $k$-spectrum kernel method are used to identify de-novo binding sites of RBPs, the identified targets are typically experimentally validated. Having higher sensitivity would ensure that the potential targets of RBP are not missed but it would be at a cost of higher false positives. The expense involved in validating these targets experimentally would determine the bias towards sensitivity or a higher PPV. If it was possible to have a two-step validation process where the first step would be a quick experiment (thus lower cost) to validate the computationally predicted targets, and only the targets that make it through this test would then be tested using the more expensive experimental method. In this scenario, higher sensitivity values would be more inclusive and would not throw away potential targets. However, if experimental validation is expensive both in terms of time and cost to carry out the experiments, biologists would prefer a lesser number of computationally predicted targets, then have a higher PPV would be preferred.
Evaluation Methodology: The existing evaluation method as discussed earlier was stringent and is indicative of sensitivity and PPV values on the lower side of the spectrum. Given a random sequence, the probability of finding a particular motif at the exact location that the motif was embedded is extremely low. However, the probability of finding a motif in a random sequence (regardless of the location) is the same as a random predictor (approximately 50 percent). To compare the results obtained by modifying the evaluation methodology where the predicted motifs are evaluated based on their presence or absence in the sequence rather than presence at a very specific embedded location. This method is less stringent and will result in higher sensitivity and PPV values.

The PySG framework was augmented to switch to this evaluation method, and results indicate that while the trend was maintained (the k-spectrum kernel did out-perform DREME in case of sensitivity and vice-versa in terms of PPV), the sensitivity and PPV values were much higher when compared to the results obtained in Section 6.3. Comparing the results in Figure 6.3 with those in Figure 6.7, the sensitivity values were 30 percentage points higher in the case of the top-25 k-mers when the sequence length was 128 nucleotides long. At the same length, DREME had 60 percentage points higher PPV value. While the PPV of the k-spectrum kernel method increased from 0.05% to 50% (when sequence length was 128 nucleotides), the PPV of the discriminative method was comparable to a random predictor. On the other hand, DREME [13] had a much higher PPV even when the sequence length was set to 1024 nucleotides. This indicates that the predicted motifs had very few false positives.

6.6 Future Work

There are several challenges yet to be tackled with regards the PySG framework.

1. Secondary structure prediction method: One obvious fact that is ignored by the current version of this framework is the inclusion of structure prediction methods such as RNAPromo [147], RNAContext [86], or MEMERIS (MEME in RNA Including Secondary structure)
Figure 6.7: Primary Sequence Motif: RBP LIN-28: Effect of varying the signal-to-noise ratio. Results obtained by using Sequence-based evaluation method. The sequence length is varied from 128 to 1024 nucleotides with the same signal embedded in each set of sequences. The sensitivity and PPV values are computed at every level of signal-to-noise ratio.

Steps to incorporate that into the framework are outlined in Chapter 5 Section 5.4.

(2) Consistent GC-content across datasets: The GC-percentage of the sequences in the dataset were listed in Section 6.3 and 6.4. As seen in Table 6.1 the GC-percentage dropped 5 percentage points as the sequence length increased from 128 nts to 1024 nts. It is unclear if the changes in GC-percentage affected the performance of the computational methods especially given that motif embedded was GA-rich. The percentage can be kept constant if the sequences were generated using a language model that is built based on a particular ATGC distribution.

(3) Vary number of sequences: The experiments carried out in this chapter demonstrated the behavior of the framework to generate sequences that varied the noise level, as well as the number of sequences that contained the embedded signal, a simple addition to this list of experiments is benchmarking the behavior of computational methods when the number of
sequences generated is varied by keeping the noise level constant and by embedding the signal in all sequences. This experiment though simplistic in nature compares the number of sequences discriminative methods need to identify strong signals present in the dataset. This information can be used when running experimental methods to generate datasets.

(4) Simulation of errors: Introduction of substitution errors, indels, and other sequencing related errors into the simulated sequences would make them mimic the output of sequencing tools.

(5) Vary composition of flanking regions: In addition, to simulate sequences that further mimic the RBP targets, the ability to embed specific nucleotide composition flanking regions could be supported. One such example is the presence of promoters in the 5’UTR flanking regions. Being able to append such regions around the embedded signal would further improve the process of sequence simulation.

(6) Evaluation Methodology: Given the variability in the results when two evaluation methodologies, a middle ground would involve, developing an evaluation technique that scores mismatches based on the composition of the embedded motif where some mismatches are higher valued than others and biological knowledge could be incorporated in this scoring scheme. This could be incorporated by representing the embedded signal as a position weight matrix.

Since the PySG framework focuses on generating tailor-made sequences, the motif embedded is always known, and hence it is easier to evaluate the motifs predicted by the computational methods. When the motif is unknown, experimental methods described in Chapter 2 are used to validate the motifs predicted by computational methods. As seen in the experiments carried out in this chapter, the top-25 $k$-mers did not always contain the embedded motif. The most important message that is highlighted by using top-25, top-50, top-100 $k$-mers is the fact that computationally best is not always the best biologically. The $k$-spectrum kernel identifies the most discriminative
\(k\)-mers and the top-25 has the highest scoring discriminative \(k\)-mers. However, they did not always contain the right biological signal (embedded motif). It is thus evident that computational tools should focus not on finding the best motif, but find a range of motifs that would capture the different motifs that are relevant for a particular biological signal.
Chapter 7

Conclusion

You solve one problem... and you solve the next one... and then the next. And if you solve enough problems, you get to come home.
— Matt Whatney, The Martian

RBP s play an important role in the regulation of numerous post-transcriptional processes [51] [134]. The binding of RBP with the target mRNA alters the fate of the bound transcript and thus controls gene expression. In addition, their interaction is known to cause several genetic diseases [111], tumor development [200], as well as their role in several cancers [1 24 152 87] making the identification of RBP targets crucial. The goal of my work in this thesis has been:

- Highlight the use of SVMs in association with the \( k \)-spectrum kernel method to identify RBP targets particularly focused on two RBPs: HuR and TTP, these proteins are known to have similar binding preferences but have opposing effects on the bound mRNA targets.

- Identify the binding sites that are specific to each RBP (HuR and TTP) as well as pinpoint the binding sites that are shared by them.

- Since RBPs also recognize the secondary structure, I wanted to explore how discriminative methods would fare when the motif contains structural elements.

In Chapter 4, the \( k \)-spectrum kernel method and DREME both discovered HuR and TTP motifs that were consistent with the published motifs; the HuR \( k \)-mers were predominantly U-rich,
and AU-rich for TTP. The $k$-spectrum kernel method identified longer $k$-mers than DREME which was helpful to identify regions around the core motif. Feature engineering further corroborated the binding sites; the results identified U/CU-rich $k$-mers for HuR binders, and AU-rich $k$-mers for TTP binders thus confirming the $k$-spectrum kernel as a method to identify RBP targets. By augmenting the feature space using domain adaption, it was possible to identify key features from the combined HuR and TTP dataset. This technique isolated $k$-mers that were specific to each RBP, as well as predicted $k$-mers that were shared by them.

In Chapter 5 in order to evaluate the performance of discriminative methods when the binding site contained structural elements, I developed a framework in Python called PySG to benchmark computational methods. The developed framework would generate both positive and negative set of sequences. The positive set of sequences is generated by embedding either primary sequence or secondary structure motifs to shuffled fragments of randomly selected regions. The negative sequences are generated by shuffling the positive set of sequences. The generation of sequences is governed by parameters that control the amount of noise (nucleotides that do not contain the embedded motif), the number of sequences that contain an embedded motif, as well as the number of sequences that can be generated.

In Chapter 6 the generated sequences are benchmarked against two discriminative methods: $k$-spectrum kernel method and DREME. The experiments conducted were simplistic and were used to demonstrate the ability of the PySG framework. The experiments run model real-world scenarios; not all the reads obtained from the sequencing technologies contain the target binding sites. This scenario simulates the signal percentage experiment. The signal-to-noise ratio experiment mimics cases such as HuR targets that are known to have longer mRNA targets whereas TTP targets are known to have very short mRNAs (See Appendix G for the histogram of lengths). The experiments demonstrated the following:

- The performance of both computational methods improved as the noise in the sequences was reduced, and the number of sequences that contained embedded signal increased.
The $k$-spectrum kernel performed better than DREME when the embedded signal was primary sequence based.

DREME out-performed the $k$-spectrum kernel method when the embedded signal was secondary sequence based. This is attributed to the identified motifs being in the IUPAC format allowing variations to the nucleotides in the motif.

DREME identified motifs that had a fewer false positive rate as compared to the $k$-mers identified by the $k$-spectrum kernel method. This was evident with higher values of PPV for DREME, and was due to the ability of DREME to identify motifs that captured the covariation that exists in secondary structure elements.

The next paragraph will summarize my contributions to the field of RBP target identification.

### 7.1 Summary

To summarize, my contributions to the field of computational biology are:

- Applied the $k$-spectrum kernel method as a solution to identify RBP targets that recognize primary sequence motifs. The kernel method was able to correctly predict the binding motifs of both HuR and TTP. This is the first time SVMs and kernel methods have been applied to identify RBP targets.

- By applying feature augmentation technique in domain adaptation to the $k$-spectrum kernel method, the binding sites specific and shared by HuR and TTP were identified. So far there aren’t computational methods that have attempted to identify RBP targets that exhibit competitive binding or tried to identify shared and specific binding motifs of two RBPs that have similar binding preferences and are known to exhibit competitive binding preferences. This work is published in the PLoS One journal under the digital object identifier (DOI): [10.1371/journal.pone.0174052](10.1371/journal.pone.0174052)
• The PySG framework developed allows computational biologists to benchmark computational methods, in order to evaluate their performance under different biological conditions, as well as when primary and secondary structure elements are involved.

This thesis explored the use of SVMs in association with the \( k \)-spectrum kernel method to identify RBP targets. The \( k \)-mers identified by this method for both HuR and TTP matched the known binding motifs of these RBPs, as well the use of domain adaptation identified shared and specific binding motifs, this approach was not suitable when the binding site had structural preferences as seen in Chapter 6. The next section will discuss future directions of this work.

### 7.2 Future Work

Chapter 4, 5, and 6 explicitly list the future directions for RBP identification, PySG framework, and benchmarking. One area in the field of RBPs that is not addressed computationally is identifying interactions between multiple RBPs as well as between RBPs and microRNAs (miRNAs). While this thesis attempts to understand the binding to targets of HuR and TTP, there is still a lot of work that needs to be done in this area. RBP proteins are also known to interact with each other and exhibit competitive binding, as seen between HuR and TTP. Binding RBPs are also known to interact with microRNAs (miRNAs) and this combinatorial binding is known to be initiated by the mRNA itself. For example, the binding of RBPs PUM1, and PUM2 causes changes to the 3’UTR of its target mRNA making it susceptible to miRNA-binding. Currently, there is no way to detect such interactions computationally. This thesis attempts to identify shared and specific RBP binding motifs of two RBPs that share binding preferences. This is the first step that needs to be further enhanced to apply to the combinatorial binding that is exhibited by RBPs.

Having explored the use of discriminative machine learning (ML) techniques such as SVMs, one additional learning method that would be an ideal candidate to explore is neural networks and more specifically deep learning neural networks. Neural networks consist of three different layers: (i) the leftmost layer is known as an input layer, (ii) the rightmost layer is called as an output
layer, and (iii) middle layer is called as the hidden layer and there can be multiple hidden layers in a given neural network. These layers are decision-making layers, and output of one layer is fed to the next layer. Each hidden layer provides a level of abstraction; for example, at each layer, a different problem can be solved. With respect to RBP targets, one layer could learn to recognize sequence-specific features, the second layer could learn to recognize structure-based features, the third layer to recognize the presence of other RBPs, and so on. Having these multiple layers of abstraction is what gives deep-learning networks an advantage and could be leveraged to identify RBP targets.

While this thesis attempted to identify specific and shared features of two RBPs, another approach that can be used in conjunction with neural networks is domain-adversarial learning. In such learning, the focus is on features that are discriminative as well as on features that are invariant to domains (have the same or very similar distributions in the source and the target domains). One example of such work by Ganin et al. [62]. They use domain-adversarial learning and use a combination of label predictor and domain classifier to make decisions in a deep-learning network. While the label classifier is optimized to minimize error on the training set, the domain classifier encourages domain-invariant features when carrying out optimizations when building a classifier.
7.3 Final Remarks

Finally, it is evident that the field of identifying targets of RBPs is gaining popularity due to increasing awareness of the role RBPs play in several diseases and post-transcriptional regulation. However, with regards to understanding their function, and their interactions, we have begun to understand the tip of the iceberg. Advances made in the field of computational biology as well as experimental methods such as iCLIP \cite{175} will facilitate further research in this field. One such example is a new class of drugs that are HuR inhibitors that are known to break the RNA-RBP interaction thus releasing the RNA and blocking HuR function as a tumor-promoting protein \cite{113}. 
Bibliography


[38] David Corcoran, Stoyan Georgiev, Neelanjan Mukherjee, Eva Gottwein, Rebecca Skalsky, Jack Keene, and Uwe Ohler. PARalyzer: definition of RNA binding sites from PAR-CLIP short-read sequence data. Genome Biology, 12(8):R79+, August 2011.


Appendix A

Preliminary Work

This appendix documents preliminary work and is provided in this thesis for completeness. Relevant background material specific to this chapter is provided at the end of the appendix.

A.1 Introduction

Chapter 2 reviewed experimental methods and their limitations identifying RBP targets. CLIP [107] is one such method that has gained popularity as it is able to identify mRNAs bound to the RBP and are known to provide 100-200 nucleotide resolution of the binding site. These methods have been used transcriptome-wide for identification of target mRNAs and target binding sites for numerous RBPs. The CLIP experiments yield data sets that are specific to a cell/tissue at a given snapshot of time (based on cell activity) and require the RBP to be active in the cell. Hence, they are unable to capture genome-wide targets of a particular RBP. Since it is impractical to perform a functional assay for a protein under every condition and cell type, my preliminary work was focused on identifying RBP targets at a genome-wide level. This chapter documents the initial computational approach and issues encountered for each approach.

A.2 Identify RBP targets: Motif Searching and Radial Basis Kernel

Most of the work done in this section was done in collaboration with Dr. Nick Farina, Jon Miller, and Paul M. Johnson as a research project for a class “Algorithms in Computational Biology”. My role in this project was to lead the computational direction, and ensure that the
information gathered about HuR binding sites from the biologist (Dr. Nick Farina) was translated correctly in the computational side.

While binding motifs had been published, these proved insufficient to identify mRNA binders from genome-wide data, since the motif was probabilistic in nature and resulted in a very high number of false positives when the motif-searching approach was used. This narrowed down the matches to 27,000 for the whole sequence data file, far too many to be a reasonable set of testable hypotheses.

The goal of the work outlined in this section was to evaluate the use of the output of motif searching as input features for machine learning to identify RBP targets from full genome data. The results showed that the motif searching approach by itself resulted in accuracy being less than a random predictor (See Table A.1), whereas when the output of motif searching was used to generate variables to feed machine learning kernel method, the accuracy improved to 78%. However, further analysis of the results (after the class project was completed) revealed that the machine learning method was picking up on signals primarily present in the negative set. This suggested that the method was not learning from the HuR binding motif. Because the negative set were transcripts that bound to IgG (negative control) at least as strongly as they bound to HuR, we concluded we likely were discriminating the non-specific binders in the negative set rather than the HuR binders.

### A.2.1 Problem Description

Predict genome-wide transcripts that would bind HuR by searching all known transcripts for the HuR binding motif, then using these results as features for an SVM.

**Input:**

- genome-wide sequence data for all known human transcripts [98]
- a probabilistic motif description for the RNA sequence to which HuR binds [41]
- positive and negative training data with transcripts known to bind HuR (positive) and known not to bind HuR (negative) [98]
Output:

- a list of transcripts predicted to bind HuR

- a model that can be used to predict whether or not a transcript will bind HuR based on features of the motif search (listed in Section A.2.2)

A.2.2 Approach

The approach used to identify RBP targets is outlined below and explained using Figure A.1.

1. translate probabilistic motif published by Silanes et al. [41] into a descriptor file for RNAMotif

2. use RNAMotif [114] to search for the motif in the positive and negative training sets of transcripts
(3) filter the returned list of transcripts in various ways:

- strength of matches
- remove overlapping motifs
- remove motifs within 17 base pairs of the previous motif, as these were considered too close to one another to be biologically relevant

(4) compute features (listed below) from the filtered matches

(5) train a radial basis kernel using the features from the positive and negative training sets

(6) run the model generated by the training sets on the features from the genome-wide transcripts to identify candidate mRNA binders

Features extracted to glean biological characteristics of HuR binding sites:

- Number of motifs per 1000 base pairs in the entire sequence
- Number of motifs per 100 base pairs in the 3’ UTR
- Number of motifs per 100 base pairs in the 5’ UTR
- Distance of the closest 3’ motif to the start of the 3’ UTR
- Distance of the closest 3’ motif to the end of the 3’ UTR
- Distance of the closest 5’ motif to the start of the 5’ UTR
- Distance of the closest 5’ motif to the end of the 5’ UTR

A.2.3 Results

Table A.1 compares the results of using motif searching to the approach where the results of motif searching were input to the radial basis kernel method. The sensitivity and specificity for the unspliced transcript was \( \frac{750}{750 + 1832} = 29.0\% \) and \( \frac{140}{140 + 1410} = 9.0\% \) respectively. For
the 3´ UTR sequences, the values were $\frac{907}{(907 + 799)} = 53.2\%$ and $\frac{1173}{(1253 + 1173)} = 48.35\%$ respectively. For the 5´ UTR sequences, the sensitivity was equal to $\frac{66}{(66 + 87)} = 43.1\%$ and specificity was equal to $\frac{1885}{(2094 + 1885)} = 47.36\%$. Using the machine learning approach, the sensitivity and specificity for the 3´ UTR sequences, was $\frac{1468}{(1468 + 410)} = 78.16\%$, $\frac{1749}{(1749 + 692)} = 71.65\%$ respectively.

<table>
<thead>
<tr>
<th>Motif-Searching Approach</th>
<th>Sequence</th>
<th>Test Outcome</th>
<th>HuR Binder</th>
<th>HuR Non-Binder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unspliced Transcript</td>
<td>Positive</td>
<td>750 (TP)</td>
<td>1410 (FP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negative</td>
<td>1832 (FN)</td>
<td>140 (TN)</td>
</tr>
<tr>
<td></td>
<td>3´ UTR</td>
<td>Positive</td>
<td>907 (TP)</td>
<td>1253 (FP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negative</td>
<td>799 (FN)</td>
<td>1173 (TN)</td>
</tr>
<tr>
<td></td>
<td>5´ UTR</td>
<td>Positive</td>
<td>66 (TP)</td>
<td>2094 (FP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negative</td>
<td>87 (FN)</td>
<td>1885 (TN)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine Learning Approach</th>
<th>Variables</th>
<th>Test Outcome</th>
<th>HuR Binder</th>
<th>HuR Non-Binder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3´ UTR</td>
<td>Positive</td>
<td>1468 (TP)</td>
<td>692 (FP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Negative</td>
<td>410 (FN)</td>
<td>1749 (TN)</td>
</tr>
</tbody>
</table>

Table A.1: Results using RNAMotif [114] to identify the presence of the predicted HuR binding motif in either the unspliced transcript, 3´ UTR, or 5´ UTR of biologically predicted HuR Binders and Non-Binders. TP = True Positive; FP = False Positive; FN = False Negative; TN = True Negative.

A.2.4 Uncompleted and Planned Directions

Add the following sets of features:

(1) Number of pairs of motifs an appropriate distance apart to indicate possible dimeric binding

(2) The distance between such a pair of motifs and the start and end of the UTRs

A.2.5 Conclusion

Summary:

(1) Scoring function written helped reduce the number of binding sites identified by RNAMotif.
(2) Using machine learning with search features initially seemed successful and predicted HuR binders from non-binders at 78% accuracy. However, key features identified by the machine learning algorithm were picked up from the negative datasets instead of positive. The learning algorithm was identifying non-specific binding of the negatives rather than HuR binding in the positive set.

Lessons learned:

(1) Even with a scoring function, motif search alone is insufficient to be a useful predictor of RBP binders.

(2) When using machine learning, it is important to realize the dataset that the algorithm is picking up the signal from.

(3) One must always understand the biological data fully.

(4) The negative set of sequences SHOULD NOT be sticky to another antibody but COULD be a set of random sequences that DO NOT contain the known binder. More generally, there should not be something common to sequences in the negative set that is not common to most other sequences.

While the developed classifier was not successful, using kernel methods to identify RBP targets was further explored by investing two other kernel methods: the $k$-spectrum kernel method, and the directed acyclic graph kernel method as discussed in the next section.

A.3 Identify RBP targets: Kernel Methods

The goal of the work outlined in this section was to evaluate the use of a string-based and structure-based kernel method to identify RBP targets. The results showed that the structure-based kernel method was unable to run on the longer HuR transcript sequences due to memory constraints of the underlying software used, whereas the string-based kernel method predicted HuR
binders only slightly better than a random predictor in some cases. The string-based kernel method revealed feature $k$-mers that matched biological binding sites of HuR.

A.3.1 Problem Description

Input:

- positive dataset: transcriptome-wide data for all known transcripts identified using PAR-CLIP [72] carried out by Kishore et al. [92]

- negative datasets: one size-matched set (290 sequences), second large random set (2470 sequences) to simulate the case where HuR binders are searched for in the entire transcriptome [167]

Output:

- a list of transcripts predicted to bind HuR

- a model that can be used to predict whether or not a transcript will bind HuR based on sequence features identified by kernel methods.

- a model that can be used to predict whether or not a transcript will bind HuR based on structural features identified by kernel methods.

- feature $k$-mers identified by the $k$-spectrum kernel method.

A.3.2 Approach

(1) generate size-matched negative datasets using non-HuR binders obtained from Biomart [167]

(2) write a python script to run the $k$-spectrum kernel method by specifying $k$-mer length

(3) generate the $k$-spectrum kernel model using positive and negative set of sequences

(4) use five-fold cross-validation to generate and validate the models
(5) integrate the directed acyclic graph (DAG) kernel method into PyML [20].

(6) generate model using five-fold cross-validation and evaluate the model on positive and negative sets of sequences.

A.3.3 Results

A.3.3.1 k-spectrum kernel method

HuR is known to have U-rich binding targets, which are also known to contain the pentamer AUUUA [163, 176]. Kishore et al. [92] listed the top ten 7-mers with the highest affinity for HuR across the different CLIP and PAR-CLIP experiments. Hence, the k-mer lengths of five and seven were tested on the k-spectrum kernel method.

<table>
<thead>
<tr>
<th></th>
<th>Balanced Success Rate</th>
<th>Sensitivity</th>
<th>PPV</th>
<th>AUC</th>
<th>ROC50</th>
</tr>
</thead>
<tbody>
<tr>
<td>3' UTR, k = 5, 290 Random</td>
<td>68.3</td>
<td>82.8</td>
<td>86.5</td>
<td>86.5</td>
<td>82.7</td>
</tr>
<tr>
<td>3' UTR, k = 5, 2469 Random</td>
<td>71.1</td>
<td>49.7</td>
<td>42</td>
<td>86.8</td>
<td>54.8</td>
</tr>
<tr>
<td>3' UTR, k = 7, 290 Random</td>
<td>58.8</td>
<td>98.9</td>
<td>54.2</td>
<td>90.6</td>
<td>87.5</td>
</tr>
<tr>
<td>3' UTR, k = 7, 2469 Random</td>
<td>60.1</td>
<td>98.9</td>
<td>12.3</td>
<td>91.2</td>
<td>75.9</td>
</tr>
<tr>
<td>5' UTR, k = 5, 290 Random</td>
<td>61.3</td>
<td>88.3</td>
<td>58.9</td>
<td>68.8</td>
<td>66.1</td>
</tr>
<tr>
<td>5' UTR, k = 5, 2469 Random</td>
<td>60.8</td>
<td>83.5</td>
<td>11.8</td>
<td>70</td>
<td>58.4</td>
</tr>
<tr>
<td>5' UTR, k = 7, 290 Random</td>
<td>55.28</td>
<td>99.6</td>
<td>54.5</td>
<td>70.4</td>
<td>67.7</td>
</tr>
<tr>
<td>5' UTR, k = 7, 2469 Random</td>
<td>65.1</td>
<td>81.7</td>
<td>13.7</td>
<td>72.3</td>
<td>59.1</td>
</tr>
</tbody>
</table>

Table A.2: k-spectrum kernel results. **Balanced success rate** is the fraction of correctly classified examples weighted by the size of the dataset and is used when the data sets are not equal size. **Sensitivity** is the fraction of examples from the positive class that is correctly classified. **Positive predictive value (PPV)** is the fraction of correct predictions made. **AUC** column refers to the area under the receiver operating characteristic (ROC) curve.

**Result analysis:** Table A.2 illustrates the results of experiments run on the data set specified by Kishore et al. [92]. The kernel method had the highest success rate when \( k = 5 \) which matches and validates the results by Sebani et al. [163] where the motif is seen as a pentamer. The success rate is much lower in the case of the 5´ UTR sequences which matches with the absence of biological evidence of HuR binders in the 5´ UTR regions.
The kernel method was unsuccessful in classifying HuR binders from non-binders as it was unable to run with the dataset provided. The lengths of the 3’UTR sequences were longer than the requirement of the Vienna RNA software [77] that was used by the kernel method. This software is used to predict minimum energy secondary structures and pair probabilities. Since the unspliced transcript lengths were longer, they were not run on this kernel. However, the kernel method attained nearly perfect discrimination for the 5’UTR data set. HuR is not known to bind in the 5’UTR regions and hence the result seems erroneous.

Sequence length comparison between HuR binders and Random data

The 3’UTR sequence length of the HuR binders was much longer compared to the non-binders. It was important to characterize the sequence length of the random genes. Figure A.2 illustrates the distribution of all the three data sets. It is clear that the HuR binders have sequences much longer than Random and Non-binder sequences. The random genes sequence lengths are somewhat longer than the non-binder genes, but it is not yet known if this is because the non-specific binders tend to be shorter than expected or only because there are (unknown) HuR binders included in the random set. This brings about a distinct point, that sequence length could be used as a differentiating factor for identifying HuR binders and also a reason why the HuR 3’UTR sequences were unable to run using the DAG kernel method [156].

Once again, the success rate was higher when the negative sequences consisted of the control set that was sticky to another antibody and published by Kishore et al. [92]. However, by using random targets as the negative set, the success rate was lower by almost 15% percent points when compared to the non-binders negative set $(k = 5)$ (See Table A.2). Both the experiments validated the use of random targets when generating the negative dataset. Growing use of PAR-CLIP experiments has resulted in the availability of higher nucleotide resolution RBP targets. In addition, the inability of computational methods to successfully run on long transcript sequences led to a change in direction of the thesis project to use PAR-CLIP reads. Thus, computer scientists
need to be ready to leave behind what they had been working on to make use of better data.

A.3.4 Uncompleted and Planned Directions

(1) Use data for TTP, an RBP that binds monomerically

(2) Incorporate 3’UTR length as a feature

(3) Create a kernel that incorporates both sequence and structure information.

A.3.5 Conclusion

Summary:

(1) DAG kernel was unable to run on longer sequences due to computational limitations of using Vienna RNA software

(2) The $k$-spectrum kernel method was able to identify binding motifs of HuR targets

Lessons learned:
Using the entire transcriptome to identify targets of RBPs was unrealistic due to nature of some 3’UTR sequences.

When the advent of new experimental methods, it is important for computer scientists to be prepared to shelf computation methods and change directions of the project.

A.4 Identify sequence and structure of RBP binding sites: RNAPromo

The goal of the work outlined in this section was to evaluate the use of the computational method based on covariance models as an approach to assess the possibility of HuR targets showing structural preferences. In order to do so, RNAPromo [147] was run on the PAR-CLIP reads (HuR-bound) which were in the 3’UTR region. RNAPromo [147] generates its own control set by shuffling the positive sequences keeping the dinucleotide distribution of the positive sequences. However, this shuffling may result in sequences that do not look like 3’UTR sequences for other reasons, so we also generated negative sequences.

A.4.1 Problem Description

Input:

- positive dataset: clusters from PAR-CLIP [72] experiments carried out by Kishore et al. [92]
- negative dataset: size-matched clusters from non-HuR transcripts [167]

Output:

- sequence and structure of HuR binding sites

A.4.2 Approach

(1) generate a length-matched negative set of sequences

(2) run the positive and negative set of sequences using RNAPromo [147]
A.4.3 Results

Figure A.3 illustrates the two types of motifs that the tool discovered. Two of the five were unstructured loop (Figure A.3(c), A.3(d)) and the other three were stem-loop structures (Figure A.3(a), A.3(b), and A.3(e)) with the loop being U-rich. The motifs identified by RNAPromo [147] had AUC scores of 96% or more which is indicative of strong presence of the HuR motif in the 3’UTR sequences.

A.4.4 Conclusion

RNAPromo [147] identified stem-loop structures as seen from the previous section. However, HuR is known to bind to single-stranded regions [121, 207]. The absence of secondary structure associated with HuR binding was further corroborated by Lebedeva et al. [100]. They used Vienna RNA package 1.8.2 [77] to computationally fold sequences within the 201 nt window centered around the 3’UTR binding sites. They used randomly chosen positions on those 3’UTRs as the control set. The base-pairing probabilities were averaged to find a substantially reduced pairing probability around the HuR anchors (peaks) but they did not find any indication of a hairpin structure. Covariance models have a tendency to search explicitly for secondary structures and this is seen in the case of HuR as well as Puβp RBPs [121, 207].

A.5 Background

This section will review the dataset used to identify RBP target sites as well as computational methods relevant to this chapter. The computational methods $k$-spectrum kernel method and RNAPromo [147] are reviewed in Chapter 2.

A.5.1 Data: Obtaining HuR binders from IP assay

Transcripts that bound HuR were identified from an immunoprecipitation assay followed by custom microarray analysis and standardization using Z-score normalization as described in [98]. Briefly, whole-cell lysates from a human cervical carcinoma cell line (HeLa) were cross-linked and
Figure A.3: HuR motifs found by RNAPromo [147] in 3′UTR region
transcripts were pulled down with either an anti-HuR antibody or IgG control. After protein digestion of the lysates, cDNA was made, radiolabeled, and hybridized to cDNA arrays. The normalized raw data were downloaded from the Gene Expression Omnibus (GEO) database\footnote{http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE1361} and analyzed using Spotfire Decision Site (TIBCO Software Inc.) as follows: an analysis of variance (ANOVA) was performed to identify significantly regulated \((p < 0.05)\) or unregulated \((p > 0.5)\) probe sets. The average \(Z\)-score in the IgG control sample was subtracted from the HuR enriched sample to identify HuR bound target sequences. A \(Z\)-score difference of 0.5 was set as the lower
bound for HuR enrichment. Those probe sets that were neither significantly regulated \((p > 0.5)\) nor enriched for HuR binding (Z-score difference −0.1 to 0.05) were used as a negative control (HuR non-binder). Any gene with probe sets in both HuR enriched and negative control datasets was removed from the negative control dataset. Figure A.4 illustrates the process carried out to acquire potential HuR target binding sites. Silanes et al. \cite{41} published the gene ids of the HuR targets which were then input to Ensembl ID \cite{167} to obtain sequences of the 3´ UTR, 5´UTR, and unspliced transcripts for those gene ids. The next section describes the approach used to identify target binding sites computationally starting with the sequences obtained.

**A.5.2 Radial Basis Kernel**

This kernel is also known as RBF kernel is a popular kernel method used in particularly with SVMs. This kernel operates on two samples \(x\) and \(x\) represented as feature vectors in some input space is defined as the squared euclidean distance between the two feature vectors.

The standard form of the RBF model is:

\[
h(x) = \sum_{n=1}^{N} w_n \exp(-\gamma \|x - x'\|^2)
\]

(A.1)

The model associated with this kernel is based on distance (hence radial) of a particular point \(x_n\) and distance of the points in its neighborhood. The key message of the RBF model is a point \(x_n\) influences nearby points more than the ones that are far away from \(x_n\).

Equation A.1 shows that for the points closer to the point \(x_n\), the point influences the points around itself, and further, the points are, the influence reduces, and it reduces as a Gaussian. The \(h(x)\) is the contribution to the point \(x\) according to the data point \(x_n\). The summation of the influences of all the data points are summed as a weight \(w_n\) and that summarizes the model. The reason it is called as radial basis function is the components that it is made up of are: (i) radial: \(\|x - x'\|\) and (ii) basis function refers to the expression: \(\exp(-\gamma \|x - x'\|^2)\).

The SVM kernel implementation of the RBF kernel is :

\[
\text{sign} \left( \sum_{\alpha_n > 0} \alpha_n y_n \exp(-\gamma \|x - x'\|^2) + b \right)
\]

(A.2)
where the sign is built-in SVMs for classification, the summation is over the support vectors, the term $\alpha_n y_n$ is the parameter weight, the label $y_n$ influences the sign of the result i.e. positive or negative, and finally the bias term is added in.

The value of $\gamma$ controls the influence of a single training example; small values of the $\gamma$ result in a large variance and thus the support vector has a greater influence. For example, support vector $x_j$ will have a greater influence on deciding the class of the vector $x_i$ even if the distance between the points is large. Conversely, if the value of $\gamma$ is large, the variance is small and the support vectors do not have as much influence.

**A.5.3 Stem kernel, Directed Acyclic graph kernel**

Sakakibara et al. [153] define the similarity feature by counting all possible non-contiguous stems of arbitrary length that two sequences have in common. The kernel function which is called the **stem kernel** calculates the inner product of two vectors in the feature space of two RNA sequences. The feature space is defined as all possible non-contiguous stems of arbitrary length of the infinite dimension. This dimension increases linearly but the amount of computation increases exponentially as the number of non-contiguous stems increases, making the computation inefficient.
To make the computation efficient, recursion is used.

**Terminology used:** \( v = a_1 a_2 ... a_n \) where \( a_i \) represents a nucleotide. \( a_k \) is denoted by \( v[k] \), a contiguous sub-sequence \( a_j ... a_k \) by \( v[j,k] \) and the length of \( v \) by \( |v| \). The empty sequence is indicated by \( \epsilon \). For base \( a \), the complementary base is denoted by \( \bar{a} \). The following equations define the recursive form of the stem kernel as proposed by Sakakibara et al. [153]:

\[
K (\epsilon, w) = K (v, \epsilon) = 1 \quad \text{for all } v, w, \quad (A.3)
\]

\[
K (va, w) = K (v, w) + \sum_{w[k] = \bar{a}} \sum_{i < j \text{ s.t } w[i] = \bar{a}, w[j] = a} K (v[k+1,|v|], w[i+1,j-1]) \quad (A.4)
\]

To complete the recursive equations, the following recursion is necessary

\[
K (v, wa) = K (v, w) + \sum_{w[k] = \bar{a}} \sum_{i < j \text{ s.t } v[i] = \bar{a}, v[j] = a} K (v[i+1,j-1], w[k+1,|w|]) \quad (A.5)
\]

The recursive equation computes the kernel value for smaller subsequences in the sequence and works its way towards larger sequences. For example \( K (va, w) \) is the sum of the kernel function \( K (v, w) \) and the additional stems that are added when the nucleotide \( a \) is considered. The intuition of this kernel method is that the more stems that two RNA sequences have in common, the more similar the sequences are to each other. This kernel uses dynamic programming and the computational complexity of this kernel \( K (v, w) \) is \( m^2 \times n^2 \), where \( m \) and \( n \) refer to the length of sequence \( v \) and \( w \) respectively. However, since the computational time and memory size required for the naive implementation of stem kernels are on the order of \( O(n^4) \), where \( n \) is the length of the input RNA sequence, applying stem kernels directly to large data sets is impractical.

Sato et al. [156] proposed a new technique based on directed acyclic graphs (DAGs) and derived from base-pairing probability matrices of RNA sequences, which significantly reduces the computational time of stem kernels. Figure A.6 illustrates the steps used to calculate the kernel. For each RNA sequence, the base pairing probability matrix is calculated using the McCaskill algorithm [120]. The Vienna RNA package [77] is used to get the expected counts using the McCaskill algorithm [120]. The next step involves building a DAG for the base-pairing probability matrix, where each vertex corresponds to a base pair whose probability is greater than a predefined threshold. Let \( G_x = (V_x, E_x) \) be the DAG for the RNA sequence \( x \), where \( V_x \) and \( E_x \) are vertices
and edges in the DAG $G_x$, respectively. For each $v_i = (k, l) \in V_x$, $(x_k, x_l)$ is a likely base pair; in other words, $P^x_{kl} \geq p^*$. Each $e_{ij} \in E_x$ is an edge from vertex $v_i$ to $v_j$. Finally, the kernel value between the two DAGs is computed representing the RNA structure information through the DAG kernel using a dynamic programming technique. The time and memory required for this algorithm are approximate of the order of $O(n^2)$. A profile-profile stem kernel is also proposed by Sato et al. [156] where if the multiple alignments of the homologous RNA sequences are available, the base-pairing probability matrices can be computed more precisely by taking the averaged sum of the individual base-pairing probability matrices based on the given multiple alignments. When multiple alignments are used, the stem kernel is called as the profile-profile stem kernel.

The experiments carried out by Sato et al. [156] demonstrate that the DAG kernel outperformed the existing methods for detection of known ncRNAs by using SVMs and kernel hierarchical clustering. When an SVM was used, the stem kernels had near perfect discrimination with a comparison to local alignment kernels - this was due to the use of secondary structure when constructing

---

**Figure A.6**: Averaged base-pairing probability matrices and DAG kernels using dynamic programming used to calculate profile-profile stem kernels for multiple alignments of RNA sequences. (a) Given multiple alignments, (b) Base-pairing probability matrices for each sequence in the multiple alignments and average base-pairing probabilities with respect to the columns of each alignment. (c) Build a DAG with averaged base-pairing probability matrix, where each vertex corresponds to a base pair whose probability is above a predefined threshold, (d) Kernel value for a pair of DAGs for multiple alignments by using DAG kernel and the dynamic programming technique.

Sato et al. [156]
the kernel function. The time and space complexity of this method are approximate of the order of $O(n^2)$. 
# DREME 4.10.2
# command:  dreme -p HuR/hur_positive.fa -n HuR/hur_control.fa -oc HuR_Output/ -mink 7 -maxk 15
# host:  USMAC1651SBHAN.local
# when:  Fri Oct 09 14:48:26 MDT 2015
# positives: 3642
# from:  HuR/hur_positive.fa (Sun Oct 04 15:20:43 MDT 2015)
# negatives: 3642
# from:  HuR/hur_control.fa (Sun Oct 04 15:20:43 MDT 2015)
MIME version 4.10.2
ALPHABET= ACGT
strands: + -
Background letter frequencies (from dataset):
A  0.278  C  0.203  G  0.207  T  0.313
MOTIF HTTTTWH DREME

<table>
<thead>
<tr>
<th>#</th>
<th>Word</th>
<th>RC Word</th>
<th>Pos</th>
<th>Neg</th>
<th>P-value</th>
<th>E-value</th>
</tr>
</thead>
<tbody>
<tr>
<td># BEST</td>
<td>HTTTTWH</td>
<td>WAAAAAD</td>
<td>2871</td>
<td>1254</td>
<td>3.2e-332</td>
<td>2.7e-326</td>
</tr>
<tr>
<td>#</td>
<td>TTTTTT</td>
<td>AAAAAA</td>
<td>1029</td>
<td>220</td>
<td>3.1e-149</td>
<td>2.6e-143</td>
</tr>
<tr>
<td>#</td>
<td>ATTTTT</td>
<td>AAAAAAT</td>
<td>850</td>
<td>198</td>
<td>8.1e-112</td>
<td>7.0e-106</td>
</tr>
<tr>
<td>#</td>
<td>TTTTTA</td>
<td>TAAAAA</td>
<td>782</td>
<td>184</td>
<td>6.5e-101</td>
<td>5.6e-095</td>
</tr>
<tr>
<td>#</td>
<td>CTTTTT</td>
<td>AAAAAAG</td>
<td>635</td>
<td>138</td>
<td>1.8e-085</td>
<td>1.5e-079</td>
</tr>
<tr>
<td>#</td>
<td>TTTTTAT</td>
<td>AAAAAA</td>
<td>601</td>
<td>148</td>
<td>1.2e-072</td>
<td>1.0e-066</td>
</tr>
</tbody>
</table>

Letter-probability matrix: length= 4 w= 7 nsites= 5073 E= 2.7e-326

0.446284 0.301202 0.000000 0.252513
0.000000 0.000000 0.000000 1.000000
0.000000 0.000000 0.000000 1.000000
0.000000 0.000000 0.000000 1.000000
0.000000 0.000000 0.000000 1.000000
0.000000 0.000000 0.000000 1.000000
0.277942 0.238912 0.000000 0.483146
0.366253 0.000000 0.000000 0.633747
Appendix c

Experimental Configurations for Benchmarking

Listing C.1: Experiment Configuration for RBP HuR

```python
targetDir: 'HuR_Parclip_SeqLength_Shuffle'
numSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
  type: 'pwmFiles'
  pwmFiles: ['HuR_ParClip1.pwm']
  signalPercent: [100]
global:
  type: 'shuffle'
  fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```

Listing C.2: Experiment Configuration for RBP TTP

```python
targetDir: 'TTP_Parclip_SeqLength_Shuffle'
numSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
  type: 'pwmFiles'
  pwmFiles: ['TTP_ParClip1.pwm']
  signalPercent: [100]
global:
  type: 'shuffle'
  fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```

Listing C.3: Experiment Configuration for RBP FUS

```python
targetDir: 'Fus_Test_SeqLength_Shuffle'
numSeq: [512]
seqLen: [128, 256, 512, 1024]
signal:
  type: 'pfmFiles'
  pfmFiles: ['FusIP_1.pfm']
  signalPercent: [100]
global:
  type: 'shuffle'
  fastaFile: '3UTR_5UTR_Transcripts_Human.fa'
```
Appendix D

Additional Benchmarking Plots

Figure D.1: Evaluate the effect of increasing sequence length on sensitivity and PPV for RBPs HuR, TTP, and FUS
Figure D.2: Evaluate the effect of increasing the number of sequences that contain the embedded signal on sensitivity and PPV for RBPs HuR, TTP, and FUS.

Figure D.3: Evaluate the effect of increasing the signal to noise ratio that contain the embedded signal of miRNA mir-101, rna structure element IRE on sensitivity and PPV.
Figure D.4: Evaluate the effect of increasing the number of sequences that contain the embedded signal of miRNA mir-101, rna structure element IRE on sensitivity and PPV.
Appendix E

Overview of existing Sequence Simulators

To validate the novelty of the sequence generator (PySG) framework that I developed, I went through the sequence simulators listed on the Genetic Simulation Resources page by the National Cancer Institute [137]. I went through each simulator, and categorized it based on the simulation it carried out.

**Population Genomics:** The simulators were categorized under this category if the sequences generated simulate the effect of genetic conditions such as mutations, gene flow, selection of certain genes on a large population. The sequences simulated are intended to study evolution so that inferences about the phylogenetics and demography of a certain population can be inferred.

**Genome Evolution:** The simulators were categorized under this category if the sequences generated contained effects of evolution that took place on the genome. The different simulations carried out were substitutions of nucleotides, gene duplication, loss of genetic content, fusion/fission of genes, rearrangement of genes.

**Mutations:** The simulators were categorized under this category if the sequences were simulated contained genetic mutations, substitutions, SNP variations, insertions, deletions, inversions and re-arrangements.

**Next-Generation Sequencing:** The simulators were categorized under this category if the sequences generated were simulating next-generation sequencing. Most of the simulators in this category also simulated empirical error models, and quality profiles. These simulators generated sequences for different types of sequencing platforms such as Illumina’s Solexa, Roche’s 454 and
Applied Biosystems’ SOLiD.

### Sequence Simulator Analysis

<table>
<thead>
<tr>
<th>Population Genomics</th>
<th>Genome Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottleNeck</td>
<td>Aladyn</td>
</tr>
<tr>
<td>BottleSim</td>
<td>ALF</td>
</tr>
<tr>
<td>CDPOP</td>
<td>BayeSSC</td>
</tr>
<tr>
<td>cosi</td>
<td>CoaSim</td>
</tr>
<tr>
<td>FEPopSim</td>
<td>DAWG</td>
</tr>
<tr>
<td>KernelPop</td>
<td>EggLib</td>
</tr>
<tr>
<td>Marlin</td>
<td>EvolSimulator</td>
</tr>
<tr>
<td>MetaPopGen</td>
<td>forqs</td>
</tr>
<tr>
<td>phenosim</td>
<td>EvolAGene</td>
</tr>
<tr>
<td>RECOAL</td>
<td>GenomePop2</td>
</tr>
<tr>
<td>Recodon</td>
<td>inverfFREGENE</td>
</tr>
<tr>
<td>SIMCOAL2</td>
<td>Nemo</td>
</tr>
<tr>
<td>SEQSIMLA</td>
<td>Mendel’s Assistant</td>
</tr>
<tr>
<td>SMARTPOP</td>
<td>PhyloSim</td>
</tr>
<tr>
<td>TreeSimJ</td>
<td>ProteinEvolver</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mutations</th>
<th>Next-Generation Sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>HapSIM</td>
<td>BEERS</td>
</tr>
<tr>
<td>MacS</td>
<td>ART</td>
</tr>
<tr>
<td>GAMETES</td>
<td>GemSIM</td>
</tr>
<tr>
<td>GWASimulator</td>
<td>Mason</td>
</tr>
<tr>
<td>HAP-GEN</td>
<td>MetaSim</td>
</tr>
<tr>
<td>FREGENE</td>
<td>prRS</td>
</tr>
<tr>
<td>SNPSim</td>
<td>rsim</td>
</tr>
<tr>
<td>SimPred</td>
<td>FLUX Simulator</td>
</tr>
<tr>
<td>SelSim</td>
<td>RNA Seq Simulator</td>
</tr>
<tr>
<td>EasyPop</td>
<td>GAMETES</td>
</tr>
<tr>
<td></td>
<td>GWASimulator</td>
</tr>
<tr>
<td></td>
<td>HAP-GEN</td>
</tr>
<tr>
<td></td>
<td>FREGENE</td>
</tr>
<tr>
<td></td>
<td>SNPSim</td>
</tr>
<tr>
<td></td>
<td>SimPred</td>
</tr>
</tbody>
</table>

**Figure E.1:** Review of existing Sequence Simulators
Appendix F

Discriminative k-mers identified by k-spectrum kernel method and DREME

F.1 k-spectrum kernel: Signal-to-noise experiment: Primary sequence motif

Listing F.1: Discriminative k-mers when the sequence length is varied

<table>
<thead>
<tr>
<th>Sequence Length: 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGGAGA, GGAGAAGT, TGGAGA, CCGGAGA, AGGAGA, GGAGA,</td>
</tr>
<tr>
<td>TTTAAA, CCCAGG, TGGAGAA, GGAGAAT, GGAGAAT, GGAGAAT, GGAGAAT,</td>
</tr>
<tr>
<td>GGAGAAA, GAGAAT, TGGAGA, GGAGA, GGAGA, GGAGA,</td>
</tr>
<tr>
<td>GGAGAAC, ACGGAGA, GAGAAT, GAGAAT,</td>
</tr>
<tr>
<td>TGGAGAA, GGAGAA, GGGAGAA,</td>
</tr>
<tr>
<td>GGAGAAAA, GAGAAT, TGGAGAA, GGAGAACC,</td>
</tr>
<tr>
<td>TTTTTT, GGGAGA, GAGAAT, GGAGAAT, GGAGAAT,</td>
</tr>
<tr>
<td>GAGGAGA, GCAGAAT, ATTTTT, TGGAGA, TGGAGA, TGGAGA,</td>
</tr>
<tr>
<td>TGGAGA, TGGAGAA, GGGAGA, GGAGAACC,</td>
</tr>
<tr>
<td>TTTTTT, TTTTTT, TTTTTT,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence Length: 256</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTTTTT, TTTTTT, AGGAGA, CCCAGG, AGGAGAAT, TGGAGA, GGAGAAT,</td>
</tr>
<tr>
<td>AAAAAAAA, AGGAGA, GGAGAACC, GGAGAAT, GGAGAACC, GGAGA,</td>
</tr>
<tr>
<td>GAGGAGA, GAGAACC, OGGGAGA, TGGAGAA, TTGTTTTT &amp; ATTTTT, TTGGTGGTTT,</td>
</tr>
<tr>
<td>GGAGAAT, GGATG, CTCAGC, CCCAGG, CAGGCT, TGIGTG, TTATTTT, TGIGTG,</td>
</tr>
<tr>
<td>TTTTTTT, CAAAAT, CAAAAT, TTTTTT, TTTTTT,</td>
</tr>
<tr>
<td>AAGGCA, CAGGACA, CGGAG, CTCAGC, AAAAAA, TTTTTT, AAAAAA, CAGGAG,</td>
</tr>
<tr>
<td>AAAAAAAA, AAAAAA &amp; TTTTTT, TTTTTT, TTTTTT,</td>
</tr>
<tr>
<td>TTTTTT, CACACAC, CACACAC, TGGGTTT, TTTTTT, CACACAC, CACACAC, CACACAC,</td>
</tr>
<tr>
<td>TTTTTT, CAAAAT, CAAAAT, AAAAAA, TTTTTT, TTTTTT,</td>
</tr>
<tr>
<td>CTTGAGAA, AATATTT, CACACACA, TTTTTT, TTTTTT, TTTTTT, CTTGAGAA,</td>
</tr>
</tbody>
</table>
F.2  DREME: Signal-to-noise experiment: Primary sequence motif

F.3  k-spectrum kernel: Signal-to-noise experiment: Secondary structure

Listing F.2: Top-25 discriminative k-mers

Table F.1: Comparing the motifs predicted by DREME \[13\] when the sequence lengths were varied with values \[128, 256, 512, 1024\] to the embedded k-mers for LIN-28 motif

F.4 k-mers identified by the DREME method: Primary sequence motif, Signal-Percentage experiment

Listing F.3: Discriminative k-mers
Appendix G

Comparison of transcript sequence lengths of HuR and TTP

**Figure G.1:** Comparison of HuR and TTP transcripts: HuR is known to bind to larger 3’UTR sequences where as TTP is known to bind to shorter 3’UTR transcripts. However, they are also known to be competitive and share binding sites.
Appendix H

RNAMotif

Figure H.1: Adapted from López de Silanes et al.[41]. (A) A probabilistic matrix of the predicted HuR motif indicates that the motif is U-rich. (C) Gives examples of representative secondary structures as predicted. We used the probabilistic matrix and the secondary structure to build the descriptor for the RNAMotif program [114].

Listing H.1: RNAMotif descriptor and scoring strategies

```python
descr
h5(len=6, pair={"g:u","u:g","u:a","a:u","g:c","c:g"}, seq="^kknddd$")
ss(minlen=5, maxlen=8, seq="^n\{0,3\}tknnw$")
h3
score
{
    bonus = 0;
    total_bonuses = 10.0;
    if (h5[1,4,1] == 'a' || h5[1,4,1] == 'u')
        bonus++;
    if (h5[1,5,1] == 'a' || h5[1,5,1] == 'u')
        bonus++;
    if (length(ss[2]) == 5) {
        if (ss[2,4,1] == 'g' || ss[2,4,1] == 'a')
            bonus++;
        if (ss[2,5,1] == 'u')
            bonus++;
    }
    if (length(ss[2]) == 6) {
        if (ss[2,5,1] == 'g' || ss[2,5,1] == 'a')
```
bonus++;  
if (ss[2,6,1] == "u")  
  bonus++;  
}  
if (length(ss[2]) == 7) {  
  if (ss[2,6,1] == "g" || ss[2,6,1] == "a")  
    bonus++;  
  if (ss[2,7,1] == "u")  
    bonus++;  
}  
if (length(ss[2]) == 8) {  
  if (ss[2,7,1] == "g" || ss[2,7,1] == "a")  
    bonus++;  
  if (ss[2,8,1] == "u")  
    bonus++;  
}  
if (h3[3,1,1] == "u")  
  bonus++;  
if (h3[3,2,1] == "g" || h3[3,2,1] == "u")  
  bonus++;  
if (h3[3,3,1] == "u")  
  bonus++;  
if (h3[3,4,1] == "a" || h3[3,4,1] == "u")  
  bonus++;  
if (h3[3,5,1] == "a" || h3[3,5,1] == "u")  
  bonus++;  
if (h3[3,6,1] == "a" || h3[3,6,1] == "u")  
  bonus++;  
}