

UNSUPERVISED FEATURE CONSTRUCTION APPROACHES FOR
BIOLOGICAL SEQUENCE CLASSIFICATION

by

KARTHIK TANGIRALA

B.Tech., Jawaharlal Nehru Technological University, India, 2009

M.S., Kansas State University, USA, 2011

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Abstract

Recent advancements in biological sciences have resulted in the availability of large amounts of sequence data (DNA and protein sequences). Biological sequence data can be annotated using machine learning techniques, but most learning algorithms require data to be represented by a vector of features. In the absence of biologically informative features, k -mers generated using a sliding window-based approach are commonly used to represent biological sequences. A larger k value typically results in better features; however, the number of k -mer features is exponential in k , and many k -mers are not informative.

Feature selection is widely used to reduce the dimensionality of the input feature space. Most feature selection techniques use feature-class dependency scores to rank the features. However, when the amount of available labeled data is small, feature selection techniques may not accurately capture feature-class dependency scores. Therefore, instead of working with all k -mers, this dissertation proposes the construction of a reduced set of informative k -mers that can be used to represent biological sequences. This work resulted in three novel unsupervised approaches to construct features:

- Burrows Wheeler Transform-based approach, that uses the sorted permutations of a given sequence to construct sequential features (subsequences) that occur multiple times in a given sequence.

- Community detection-based approach, that uses a community detection algorithm to group similar subsequences into communities and refines the communities to form motifs (group of similar subsequences). Motifs obtained using the community detection-based approach satisfy the ZOMOPS constraint (Zero, One or Multiple Occurrences of a Motif Per Sequence). All possible unique subsequences of the obtained motifs are then used as features to represent the sequences.
- Hybrid-based approach, that combines the Burrows Wheeler Transform-based approach and the community detection-based approach to allow certain mismatches to the features constructed using the Burrows Wheeler Transform-based approach.

To evaluate the predictive power of the features constructed using the proposed approaches, experiments were conducted in three learning scenarios: supervised, semi-supervised, and domain adaptation for both nucleotide and protein sequence classification problems. The performance of classifiers learned using features generated with the proposed approaches was compared with the performance of the classifiers learned using k -mers (with feature selection) and feature hashing (another unsupervised dimensionality reduction technique). Experimental results from the three learning scenarios showed that features constructed with the proposed approaches were typically more informative than k -mers and feature hashing.

UNSUPERVISED FEATURE CONSTRUCTION APPROACHES FOR
BIOLOGICAL SEQUENCE CLASSIFICATION

by

KARTHIK TANGIRALA

B.Tech., Jawaharlal Nehru Technological University, India, 2009

M.S., Kansas State University, USA, 2011

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
Doina Caragea

Copyright

KARTHIK TANGIRALA

2015

Abstract

Recent advancements in biological sciences have resulted in the availability of large amounts of sequence data (DNA and protein sequences). Biological sequence data can be annotated using machine learning techniques, but most learning algorithms require data to be represented by a vector of features. In the absence of biologically informative features, k -mers generated using a sliding window-based approach are commonly used to represent biological sequences. A larger k value typically results in better features; however, the number of k -mer features is exponential in k , and many k -mers are not informative.

Feature selection is widely used to reduce the dimensionality of the input feature space. Most feature selection techniques use feature-class dependency scores to rank the features. However, when the amount of available labeled data is small, feature selection techniques may not accurately capture feature-class dependency scores. Therefore, instead of working with all k -mers, this dissertation proposes the construction of a reduced set of informative k -mers that can be used to represent biological sequences. This work resulted in three novel unsupervised approaches to construct features:

- Burrows Wheeler Transform-based approach, that uses the sorted permutations of a given sequence to construct sequential features (subsequences) that occur multiple times in a given sequence.

- Community detection-based approach, that uses a community detection algorithm to group similar subsequences into communities and refines the communities to form motifs (group of similar subsequences). Motifs obtained using the community detection-based approach satisfy the ZOMOPS constraint (Zero, One or Multiple Occurrences of a Motif Per Sequence). All possible unique subsequences of the obtained motifs are then used as features to represent the sequences.
- Hybrid-based approach, that combines the Burrows Wheeler Transform-based approach and the community detection-based approach to allow certain mismatches to the features constructed using the Burrows Wheeler Transform-based approach.

To evaluate the predictive power of the features constructed using the proposed approaches, experiments were conducted in three learning scenarios: supervised, semi-supervised, and domain adaptation for both nucleotide and protein sequence classification problems. The performance of classifiers learned using features generated with the proposed approaches was compared with the performance of the classifiers learned using k -mers (with feature selection) and feature hashing (another unsupervised dimensionality reduction technique). Experimental results from the three learning scenarios showed that features constructed with the proposed approaches were typically more informative than k -mers and feature hashing.

Table of Contents

Table of Contents	viii
List of Figures	xii
List of Tables	xv
Acknowledgements	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Overview of Proposed Approaches	3
1.2.1 Burrows Wheeler Transform Approach	3
1.2.2 Community Detection Approach	4
1.2.3 Hybrid Approach	5
1.3 Biological Problems Addressed	5
1.4 Research Questions	6
1.5 Contributions	7
1.6 Outline	8
2 Background	10
2.1 Biological Background	10
2.1.1 Genes and Proteins	10
2.1.2 Alternative Splicing and Protein Localization	11
2.2 Machine Learning Background	14

2.2.1	Learning Frameworks	14
2.2.2	Feature Representation and Dimensionality Reduction	22
3	Related Work	27
3.1	Burrows Wheeler Transform in Bioinformatics	27
3.2	Community Detection in Bioinformatics	28
3.3	Sliding Window Approach in Bioinformatics	30
3.4	Feature Selection	30
3.5	Sequential Pattern Mining Algorithms	31
4	Proposed Feature Construction Approaches	34
4.1	Burrows Wheeler Transform Approach	35
4.1.1	Burrows Wheeler Transform Preliminaries	35
4.1.2	Feature Construction based on Burrows Wheeler Transform	35
4.2	Community Detection Approach	40
4.2.1	Community Detection Preliminaries	40
4.2.2	Identifying Motifs Using Community Detection - TFBSGroup	43
4.2.3	Feature Construction for Large Nucleotide Sequence Datasets	45
4.2.4	Feature Construction for Protein Sequence Datasets	48
4.3	Hybrid Approach	50
4.3.1	Motivation	50
4.3.2	Feature Construction Using the Hybrid Approach	51
5	Experimental Setup	54
5.1	Datasets	54
5.1.1	Alternative Splicing Datasets	54
5.1.2	Protein Localization Datasets	56

5.2	Research Questions	58
5.3	Experimental Setup: 5-fold Cross-Validation	59
5.4	Learning Algorithms and Other Experimental Details	62
6	Burrows Wheeler Transform Approach: Experiments and Results	64
6.1	Research Questions	64
6.2	Parameters and Experiments	66
6.2.1	Default Parameters	66
6.2.2	Experiments	67
6.3	Results	69
6.3.1	Dimensionality Comparison	69
6.3.2	Supervised Scenario: <i>b</i> -mers <i>versus</i> <i>k</i> -mers	69
6.3.3	Semi-supervised Scenario: <i>b</i> -mers <i>versus</i> <i>k</i> -mers	73
6.3.4	Domain Adaptation Scenario: <i>b</i> -mers <i>versus</i> <i>k</i> -mers	74
7	Community Detection Approach: Experiments and Results	80
7.1	Research Questions	80
7.2	Parameters and Experiments	83
7.2.1	Default Parameters	83
7.2.2	Experiments	84
7.3	Results	87
7.3.1	Dimensionality Comparison	87
7.3.2	Supervised Scenario: <i>c</i> -mers <i>versus</i> <i>k</i> -mers	87
7.3.3	Semi-supervised Scenario: <i>c</i> -mers <i>versus</i> <i>k</i> -mers	91
7.3.4	Domain Adaptation Scenario: <i>c</i> -mers <i>vs</i> <i>k</i> -mers	92
7.3.5	Varying the Number of Motifs	92
7.3.6	Varying the Number of Samples and Sample Size	97

7.3.7	Varying the Number of Mismatches and Hamming Distance	99
7.3.8	Varying the Substitution Score Threshold	101
8	Hybrid Approach: Experiments and Results	103
8.1	Research Questions	103
8.2	Parameters and Experiments	104
8.2.1	Default Parameters	104
8.2.2	Experiments	106
8.3	Results	109
8.3.1	Dimensionality Comparison	109
8.3.2	Supervised Learning Scenario	109
8.3.3	Semi-supervised Learning	119
8.3.4	Domain Adaptation Scenario	120
9	Conclusion and Future Work	127
9.1	Conclusion	127
9.1.1	Contributions	127
9.1.2	Merits	128
9.1.3	Limitations	129
9.2	Future Work	130
	Bibliography	132

List of Figures

2.1	Simplified gene structure	11
2.2	Central dogma of molecular biology	12
2.3	Splicing phase of central dogma	12
2.4	Alternatively spliced and constitutive exons	13
2.5	Abstract working of the supervised learning scenario.	17
2.6	Abstract working of the semi-supervised learning scenario.	18
2.7	Abstract working of the domain adaptation scenario.	22
4.1	Feature construction with the Burrows Wheeler Transform approach.	36
4.2	Communities of a network	41
4.3	Construction of a N -partite graph of k -mers for two nucleotide sequences	44
4.4	Sequential feature construction with community detection approach	46
4.5	Sample motif identified using the community detection algorithm	47
4.6	PAM30 substitution matrix	49
4.7	Sequential feature construction with hybrid approach	51
4.8	Using b -mers to reduce network nodes in HBA approach.	52
5.1	General preprocessing of alternative splicing sequences (exon triplets)	56
5.2	The 5-fold cross-validation setting for conducting experiments	60
5.3	The 5-fold cross-validation setting for the supervised learning scenario	60
5.4	The 5-fold cross-validation setting for the semi-supervised learning scenario	61
5.5	The 5-fold cross-validation setting for the domain adaptation scenario	62

6.1	Evaluation of Burrows Wheeler Transform-based features in the supervised learning scenario, using the NBM algorithm	72
6.2	Evaluation of Burrows Wheeler Transform-based features in the semi-supervised learning scenario, using the self-training algorithm	76
6.3	Evaluation of Burrows Wheeler Transform-based features in the semi-supervised learning scenario, using the co-training algorithm	77
6.4	Evaluation of Burrows Wheeler Transform-based features in the domain adaptation scenario, using NBM for domain adaptation algorithm	78
7.1	Evaluation of community detection-based features in the supervised learning scenario, using the NBM algorithm	89
7.2	Evaluation of community detection-based features in the semi-supervised learning scenario, using the self-training algorithm	94
7.3	Evaluation of community detection-based features in the semi-supervised learning scenario, using the co-training algorithm	95
7.4	Evaluation of community detection-based features in the domain adaptation scenario, using NBM for domain adaptation algorithm	96
7.5	Variation of the AUC values with the number of top motifs selected when using community detection-based approach	98
7.6	Variation of the AUC values with sample size (S) and number of samples (R) when using community detection-based approach	100
8.1	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the supervised learning scenario, using the NBM algorithm	113
8.2	Evaluation of b -mers, c -mers and c_b -mers in the supervised learning scenario, with NBM algorithm	114

8.3	Evaluation of b -mers and h -mers in the supervised learning scenario, with NBM algorithm	115
8.4	Evaluation of c_b -mers and h -mers in the supervised learning scenario, with NBM algorithm	116
8.5	Evaluation of h -mers and u -mers in the supervised learning scenario, with NBM algorithm	117
8.6	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the semi-supervised learning scenario, using the self-training algorithm	121
8.7	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the semi-supervised learning scenario, using the co-training algorithm	122
8.8	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the domain adaptation learning scenario, using NBM for domain adaptation algorithm	125

List of Tables

6.1	Dimensionality comparison: Burrows Wheeler Transform-based approach . . .	69
6.2	Evaluation of Burrows Wheeler Transform-based features in the supervised learning scenario	71
6.3	Evaluation of Burrows Wheeler Transform-based features in the semi-supervised learning scenario	75
6.4	Evaluation of Burrows Wheeler Transform-based features in the domain adaptation learning scenario	79
7.1	Dimensionality comparison: Community detection-based approach	87
7.2	Evaluation of community detection-based features in the supervised learning scenario	88
7.3	Evaluation of community detection-based features in the semi-supervised learning scenario	93
7.4	Evaluation of community detection-based features in the domain adaptation scenario	97
7.5	AUC values obtained with NBM classifiers learned in the supervised scenario, when varying the number of motifs, t , selected to construct features in the community detection-based approach	98
7.6	AUC values obtained with NBM classifiers learned in the supervised scenario, when varying the number of mismatches, d , in the community detection-based approach.	101

7.7	AUC values obtained with NBM classifiers learned in the supervised scenario, when varying the maximum Hamming distance, x , in the community detection-based approach.	101
7.8	AUC values obtained with NBM classifiers learned in the supervised scenario, when varying the minimum substitution score, s , in the community detection-based approach.	102
8.1	Dimensionality comparison: BWT, CDA and HBA approaches versus union of BWT and CDA-based features	110
8.2	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the supervised learning scenario	118
8.3	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the semi-supervised learning scenario, when self-training is used as the semi-supervised classifier.	123
8.4	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the semi-supervised learning scenario, when co-training is used as the semi-supervised classifier.	124
8.5	Evaluation of b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers in the domain adaptation learning scenario	126

Acknowledgments

The following dissertation, while an individual work, would not have been possible without the help and supervision of several people. I am greatly thankful to all the people who have helped and inspired me during my Ph.D years.

First and foremost, I am greatly indebted to my adviser, Dr. Doina Caragea. It has been an honor to be her student for both my M.S. and Ph.D. This dissertation would not have been completed without her patience and steadfast encouragement. I appreciate all her contributions of time and ideas that helped me have six productive years at Kansas State University and a great experience. Her insightful comments and constructive criticisms at different stages of my research were thought-provoking and they helped me focus my ideas. Dr. Caragea was a fabulous advisor: sharp, perceptive, and mindful of the things that truly matter. I am indebted to her for her continuous encouragement and guidance. Thank you does not seem sufficient but it is said with appreciation and respect.

I am grateful to be a student of Dr. Susan J. Brown and to have her as my Ph.D. committee member. Her knowledge and ideas in the field of bioinformatics motivated me during my early stages of education at Kansas State University. I am also thankful to my M.S. committee members, Dr. Torben Amtoft and Dr. Mitch Neilsen, for their support and guidance right from the beginning of my years at Kansas State University.

I am also thankful to the former and current staff at Department of Computing and Information Sciences, Kansas State University, for their various forms of support during my graduate study.

I would like to thank Ana Stanescu and Nic Herndon for their collaborative work and valuable discussions related to research. I would like to thank all my friends for their wonderful support during all these years. I greatly value their friendship and deeply appreciate their belief in me.

Most importantly, none of this would have been possible without the love and patience of my family. My deepest gratitude to my parents, Mr. V. G. Krishna Murthy Tangirala and Mrs. V. Satya Kumari Kambhampati, to my brother-in-law, Mr. V. S. Harinarayana Vemu, and to my sister, Mrs. Harika Vemu, for their love and support at every stage of my life. It is only because of their motivation and encouragement that I am able to complete my Ph.D.

At last, I want to acknowledge the importance of the Beocat Research Cluster to my work. The computing for this dissertation was performed on the Beocat Research Cluster at Kansas State University, which is funded in part by grants MRI-1126709, CC-NIE-1341026, MRI-1429316, CC-IIE-1440548.

Chapter 1

Introduction

1.1 Motivation

Next-generation sequencing technologies have led to the availability of large amounts of biological sequence data (i.e., raw data, such as nucleotide sequences, and derived data, such as protein sequences), resulting in challenging sequential data annotation. Machine learning is commonly used to address classification problems in the field of bioinformatics, especially problems that help annotate biological sequences. However, learning algorithms require data to be represented as vectors of features. In general, the more informative the features are, the better the classifiers trained from the respective data. When available, biologically informative features (e.g., known DNA motifs or protein domains) are used to represent sequences. For most problems however, biologically informative motifs or domains are not readily available. In the absence of biologically informative motifs, the sliding window approach is commonly used to generate sequential features, referred to as k -mers.

In the sliding window approach, a window of size, k , is traversed across all sequences. The fragment of the sequence within the window is captured, and all possible unique fragments form a set of k -mers. Motifs of variable-length are believed to carry better information than motifs of fixed length. The size of the window, k , is varied in order to generate variable length

k -mers. Disadvantages of variable-length k -mers include high dimensionality of feature space (the number of features used to represent the sequences). For large datasets, the number of k -mers constructed is exponential in k . High-dimensional feature spaces increase learning time (time taken to learn a classifier from the data) and classification time (time taken by the classifier to classify new data) of the algorithm by a large extent. In addition, certain features among the constructed k -mers may not be informative, potentially acting as noise and misleading the classifier.

Feature selection techniques are commonly used to reduce dimensionality of input feature space while retaining a majority of the informative features. Most feature selection techniques use available labeled data to estimate feature-class dependencies of the features. The features are then sorted and filtered based on corresponding feature-class dependency scores. Feature selection can be applied in supervised learning scenario (large amounts of labeled data are used in the learning process), semi-supervised learning (SSL) scenario (small amounts of labeled and large amounts of unlabeled data are used in the learning process), and domain adaptation scenario (large amounts of labeled data from a source domain as well as small amounts of labeled data and large amounts of unlabeled data from the target domain are used to classify new unseen data of the target domain). However, in SSL or domain adaptation scenarios, as the amount of available labeled data is small, feature selection may not accurately capture the feature-class dependency scores. Therefore, alternative methods to generate a reduced set of informative features can presumably benefit semi-supervised learning and domain adaptation algorithms.

The work in this dissertation focuses on generating a low-dimensional informative feature set to represent biological sequences in the context of biological sequence classification using learning algorithms. A specific objective of this work is to generate a reduced set of informative variable-length k -mers, without using the class labels of the sequences (in an unsupervised manner).

1.2 Overview of Proposed Approaches

The premise of this work is to generate a low-dimensional informative feature set that could improve the performance of the learning algorithms in terms of run time and accuracy. Therefore, this work proposes three novel approaches: the first approach uses Burrows Wheeler Transform (BWT), a context-based transformation of a sequence; the second uses a community detection algorithm (CDA); and the third approach is a hybrid approach (HBA), a combination of the BWT and CDA-based approaches. The proposed approaches are unsupervised (do not take into account class labels of sequences) and can therefore use knowledge from labeled and unlabeled data in SSL or domain adaptation scenarios to generate a low-dimensional feature set, as opposed to the feature selection technique which uses only small amounts of available labeled data. This work investigates performance of the proposed approaches in supervised, semi-supervised, and domain adaptation scenarios.

The following sections provide a brief overview of the three proposed approaches (BWT, CDA, and HBA) in order to generate a reduced set of informative k -mers. As these approaches do not take into account class labels of the sequences, they have the potential to provide significant help to semi-supervised and domain adaptation approaches.

1.2.1 Burrows Wheeler Transform Approach

For biological sequences, filtering sequential features based on the frequency of occurrence is a simple and traditional approach to reduce dimensionality of the feature space. The principle of filtering k -mers that occur multiple times in a given sequence motivated the use of Burrows Wheeler Transform (BWT) to generate a reduced set of k -mers. BWT was first introduced by [Burrows and Wheeler \[1994\]](#) to address data compression. The ability of BWT to efficiently identify multiple occurrences of a sequence fragment generated significant interest among researchers, especially in the field of bioinformatics. Characters in the BWT of a sequence are grouped based on similarity of corresponding suffixes in the

original sequence. The work in this dissertation exploits this grouping of prefixes based on lexicographically similar suffixes, in order to generate variable-length features that occur multiple times in at least one sequence. In addition to filtering sequential features based on the frequency of occurrence, the BWT-based approach also takes into account other properties, such as suffix information and length, in order to retain most informative features while filtering out uninformative features.

1.2.2 Community Detection Approach

Motifs that are informative with respect to a biological problem (e.g., alternative splicing events, splice sites, protein localization) may occur at various locations (across different sequences) with certain mismatches. Therefore, identifying those k -mers that occur in several sequences with a small number of mismatches might result in an informative set of features. The work in this dissertation, proposes a novel approach of using a community detection algorithm to identify sequential features with mismatches.

A community is a sub-network with nodes that are highly/densely connected compared to other network nodes. A community reflects a group of closely associated nodes. Communities in a network identify structural and topological properties of a network. A motif is a set of closely associated subsequences of certain length with mismatches. This dissertation, investigated the potential use of communities to identify motifs, by defining nodes as subsequences of certain length and communities as sets of closely associated subsequences. In the process of identifying motifs through communities, a community detection algorithm to initially identify all possible communities in a network of subsequences was used. Each community, a set of subsequences, was further refined to form a motif. As most of the community detection algorithms are time consuming, a fast and effective technique to generate community detection-based features to represent sequences for classification purposes was proposed in this work.

1.2.3 Hybrid Approach

The BWT-based approach generates mostly non-overlapping subsequences that occur multiple times in at least one sequence. Contrarily, the CDA-based approach identifies subsequences that occur across different sequences with certain mismatches. Under the assumption that BWT features that occur across different sequences with mismatches are potentially more informative than the BWT and CDA features alone, a hybrid approach that combines the BWT and CDA-based approaches was proposed to construct sequential features. A pipeline in which features obtained using the BWT-based approach were redirected to the CDA-based approach was created, thereby identifying BWT features that occur across different sequences with mismatches.

1.3 Biological Problems Addressed

This work primarily compares the performance of features constructed using the proposed approaches with k -mers. Experiments were conducted on nucleotide and protein sequences for the following sequence classification problems:

- **Prediction of alternative splicing events:** Alternative splicing, a process that occurs during gene regulation, is responsible for protein diversity. Five major types of alternative splicing events exist: exon skipping, intron retention, alternative donor site, alternative acceptor site and mutually exclusive exons. This work specifically addresses the problem of classifying exons (DNA sequences) as either alternatively spliced or constitutive.
- **Prediction of protein localization:** Protein localization prediction is the process of predicting where the protein resides in the cell. Protein subcellular localization identification, essential for genome annotation, also plays an important role in understanding protein function. Experiments were conducted on four protein subcellular

localization datasets.

1.4 Research Questions

The following research questions were addressed in this dissertation:

- **How informative are the features constructed by exploiting properties of BWT?**

BWT produces a context dependent permutation of an input sequence (set of characters) such that characters adjoining similar contexts are grouped together. Therefore, any contiguous occurrence of a character in the BWT of a sequence can reflect an associated subsequence occurring multiple times across the sequence. This work investigates the potential use of this BWT property to generate a low-dimensional informative sequential feature set for various learning scenarios (supervised, semi-supervised, and domain adaptation).

- **How informative are the features constructed using CDA-based approach?**

This work investigates if communities can be related to motifs, used to generate low-dimensional informative features. Because most of the community detection algorithms are time-consuming, this work also proposes a fast and effective technique to generate community detection-based features.

- **Can a hybrid approach between BWT and CDA-based approaches result in even better features as compared to features obtained from each approach individually?**

The BWT-based approach, when combined with the CDA-based approach, generates BWT features with certain mismatches that occur across various sequences. This work investigates the predictive power of the features constructed using the hybrid approach to classify biological sequences.

1.5 Contributions

The major published contributions of this dissertation include:

- **Generating Features Using Burrows Wheeler Transformation for Biological Sequences** [[Tangirala and Caragea, 2014b](#)]
- **Semi-supervised Classification of Protein Sequences Using Burrows Wheeler Transformation-based Features** [[Tangirala and Caragea, 2014c](#)]
- **Predicting Protein Localization Using a Domain Adaptation Naïve Bayes Classifier with Burrows Wheeler Transform Features** [[Herndon et al., 2014](#)]
- **Community Detection-based Features for Sequence Classification** [[Tangirala and Caragea, 2014a](#)]
- **Community Detection-based Feature Construction for Protein Sequence Classification** [[Tangirala et al., 2015](#)]

Other anticipated contributions of this dissertation include:

- **A Hybrid Approach to Construct Sequential Features for Biological Sequence Classification Problems**

1.6 Outline

The rest of the dissertation is organized as follows:

- **Chapter 2:** A brief introduction to the two biological problems addressed in this dissertation followed by machine learning preliminaries are presented. Furthermore, details about sliding window approach used to generate sequential features, k -mers, to represent biological sequences and details of feature selection and feature hashing techniques also are discussed.
- **Chapter 3:** Reviews previous work on the applications of BWT and community detection in bioinformatics. Furthermore, a review of various feature selection techniques, approaches that used features constructed using the sliding window approach, and approaches designed to construct or identify patterns and motifs from biological sequences are presented as well.
- **Chapter 4:** A detailed explanation of the three proposed approaches (BWT, CDA and HBA) to construct low-dimensional informative sequential features.
- **Chapter 5:** Description of the datasets corresponding to the alternative splicing and protein localization problems, followed by experimental setup to evaluate the predictive power of the proposed features in different learning scenarios are discussed.
- **Chapter 6:** Outlines specific research questions, and presents corresponding experiments and associated results to evaluate the BWT-based approach. A detailed analysis of the results suggests that the BWT-based approach is successful in reducing the dimensionality of k -mers, while retaining most informative features.
- **Chapter 7:** Outlines specific research questions, and presents corresponding experiments and associated results to evaluate the CDA-based approach. A thorough investigation of the effect of different parameters on the predictive power of features

constructed using the CDA-based approach is conducted. A detailed analysis of the results suggests that the CDA-based approach is successful in reducing the dimensionality of k -mers, while retaining most informative features.

- **Chapter 8:** Outlines specific research questions, and presents the corresponding experiments and associated results to evaluate the HBA approach. Experiments have been primarily conducted to compare proposed approaches with each other and with feature hashing.
- **Chapter 9:** A summary of the proposed approaches to construct sequential features, list of contributions, a set of limitations, and several directions for future work are discussed.

Chapter 2

Background

2.1 Biological Background

2.1.1 Genes and Proteins

A *gene* is the hereditary unit of every living organism [Pennisi, 2007; Gerstein et al., 2007; Pearson, 2006]. Genes are segments of DNA (Deoxyribonucleic acid) consisting of two complementary strands held together by hydrogen bonds. A DNA molecule is made up of four different nucleotides: guanine (G), cytosine (C), adenine (A), and thymine (T). Nucleotide G pairs with C, and A with T. In eukaryotes, a gene consists of exons that represent protein coding information and intervening sequences called introns. The beginning of an intron is marked by a *donor site*, and the end of the intron is marked by an *acceptor site*, together, referred to as splice sites. Figure 2.1 shows a simplified picture of the gene structure.

Genes are responsible for growth and development, and they play a crucial role in protein synthesis. Protein synthesis consists of several stages, of which gene transcription is the first stage. During transcription, RNA polymerase, an enzyme that produces the primary transcript Ribonucleic acid (RNA), traverses across the gene to generate pre-mRNA, which is the unprocessed mRNA. Transcription begins at the transcription start site and ends

at the transcription stop site (shown in Figure 2.1). The pre-mRNA is then transformed into mRNA during splicing, a process that occurs in between transcription and translation phases. During splicing, pre-mRNA is modified so that introns are skipped and exons are joined. Splicing is carried out by a complex called spliceosome, which contains activity that cleaves and joins the RNA transcript. The flanked intron forms a loop structure, called a lariat, involving the adenine base. The lariat is later subsequently degraded. Finally, during the translation phase, the mRNA is transformed into a protein. Translation begins at the translation start site and ends at the translation stop site. The entire process is known as the Central Dogma of Molecular Biology, as shown in Figure 2.2. The mRNA contains protein coding information. Figure 2.3 details the splicing step in the protein synthesis process. In this picture, the pre-mRNA corresponding to a gene consisting of four exons was transformed into mRNA; all exons were retained, and all introns were skipped.

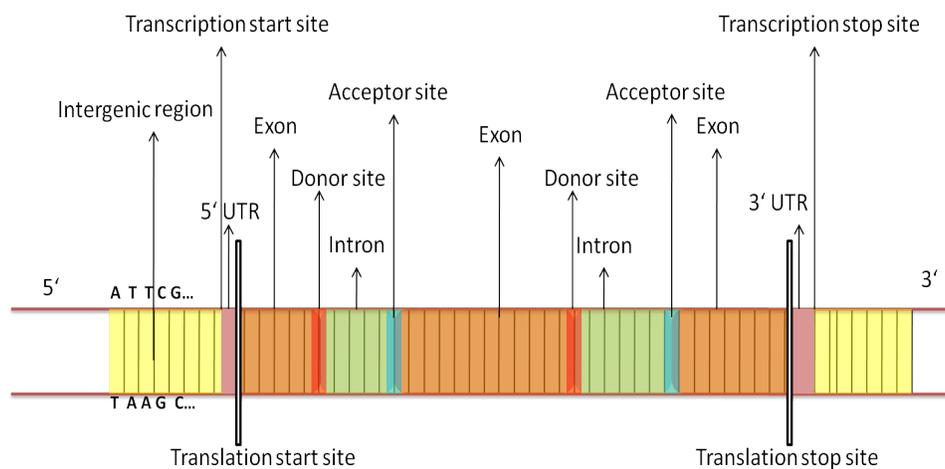


Figure 2.1: *Simplified gene structure: components of a gene relevant to this work are highlighted.*

2.1.2 Alternative Splicing and Protein Localization

This work addresses two sequence classification problems:

- Alternative splicing events in genes (nucleotide sequence classification)

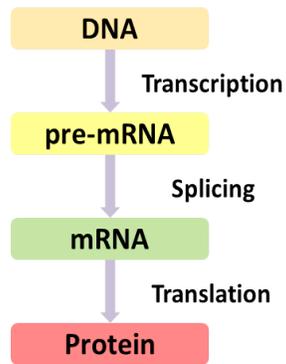


Figure 2.2: *Central dogma of molecular biology: main steps involved in protein synthesis.*

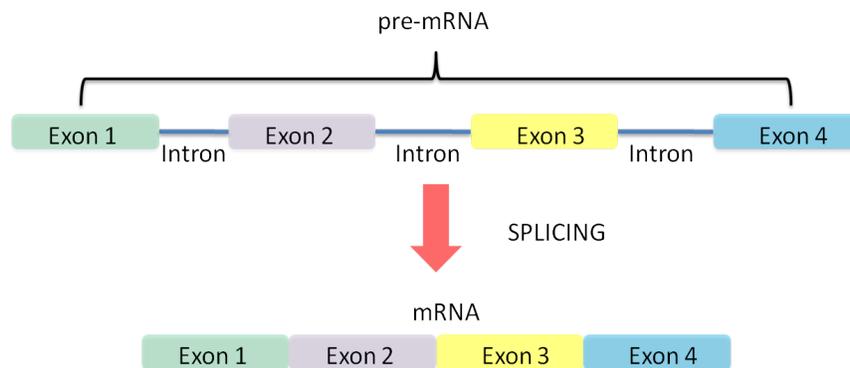


Figure 2.3: *Splicing phase of central dogma in which the pre-mRNA is transformed into mRNA.*

- Protein localization (protein sequence classification)

Alternative Splicing

One gene was historically believed to correspond to one protein. However, the discovery of alternative splicing provided an explanation for protein diversity [Black, 2003].

Alternative splicing is a mechanism responsible for the formation of multiple proteins from a single gene, meaning that several mRNA *isoforms* can be generated from the pre-mRNA corresponding to a gene. A skipped exon, a retained intron, alternative 5' donors, alternative 3' acceptors and mutual exclusive exons, as discussed by Black [2003], are possible alternative splicing events.

Figure 2.4 shows an example of an alternatively spliced exon, specifically, Exon 3 in Figure 2.4 is retained in one transcript and skipped in another transcript, thereby being alternatively spliced. Exons 1, 2, and 4 appear in both transcripts, so they are constitutive exons. Recent studies have found that approximately 95% of human genes are alternatively spliced [Pan et al., 2008]. As alternative splicing events contribute significantly to protein diversity, identifying such events is important.

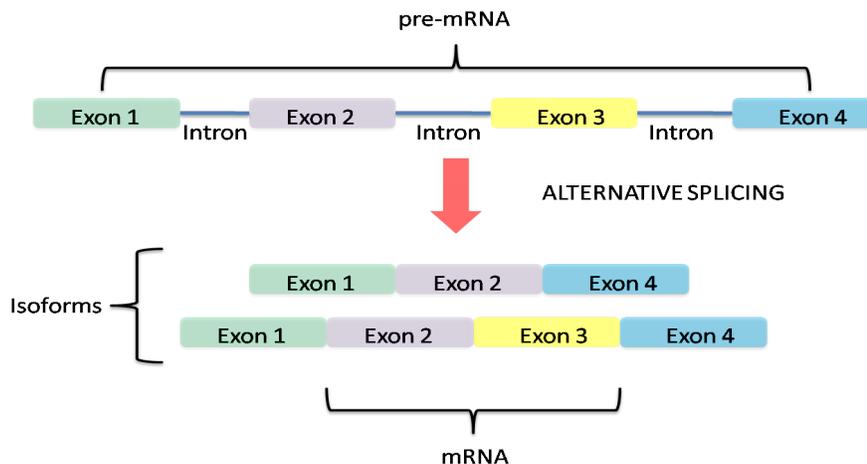


Figure 2.4: *Alternatively spliced and constitutive exons: Exon 3 is skipped in the first transcript and retained in the second transcript, therefore, alternatively spliced. Exons 1, 2, and 4 are constitutive exons because they are retained in both transcripts.*

Protein Localization

Most proteins in eukaryotes are encoded in the nuclear genome, synthesized in the cytosol, and sorted before they reach their final destination. For prokaryotes, proteins are synthesized in cytoplasm; some of the synthesized proteins must be targeted to other locations. Proteins must be localized at their respective subcellular location in order to perform their functions. Identification of protein's subcellular localization is an important step in many analyses because it may provide evidence regarding protein function. Identifying protein localization can also benefit the study of a particular protein. In the case of surface-exposed and secreted proteins, protein localization identification can also benefit the process of identifying drug targets [Bellier et al., 2013; Mora et al., 2003; Allan and Wren, 2003].

However, experimental determination of protein subcellular localization is a costly process in terms of time and work [Rey et al., 2005]. Increased availability of protein localization datasets motivated researchers to use advanced computational tools, including machine learning, for classifying protein sequences based on localization.

2.2 Machine Learning Background

Machine learning [Mitchell, 1997] is a branch of artificial intelligence focused on the design and development of algorithms for learning classifiers from data. Machine learning algorithms are primarily designed to address classification problems less expensively, including DNA or protein sequence classification problems, at several orders of magnitude faster than human experts who can potentially classify sequences.

2.2.1 Learning Frameworks

Traditional machine learning algorithms use labeled data (class labels of the data are known) to train a classifier, and use the model to predict new unseen data. This framework is also referred to as the supervised learning framework. In real world, available data can be

labeled or unlabeled (class labels of the data are unknown). For several problems, especially in biology, labeling data is a costly process. Therefore, in addition to using available labeled data, machine learning algorithms that utilize knowledge from available unlabeled data or knowledge of labeled data from a different domain could benefit the classification process. Various learning frameworks have been proposed to best fit different assumptions regarding availability of labeled and unlabeled data. The following three learning frameworks were considered in this work:

- Supervised learning
- Semi-supervised learning
- Domain adaptation

A majority of machine learning algorithms require data to be represented as a vector of features with each feature carrying a certain degree of information about a class. Classification accuracy of a classifier is largely affected by the quality of the features. The more informative the features used to represent the data are, the better the classifier trained from the respective data. Feature representation of a data sample is referred to as an *instance*. An instance is an n dimensional vector corresponding to a particular sample (biological sequence in this work), where n is the total number of features used to represent the data. Following is additional background information of the three learning frameworks used in this work.

Supervised Learning

Supervised learning utilizes labeled data to train a classifier and then the classifier is used to predict new unseen test data. Performance of the supervised classifier is highly dependent on the amount of labeled data. When the amount of available labeled data is small, supervised classifier, may result in poor performance. Figure 2.5 shows an abstract working of a supervised learning algorithm. Support Vector Machines (SVM) [Cortes and Vapnik,

1995], Random Forests (RF) [Breiman, 2001] and Naïve Bayes (NB) [Kibriya et al., 2004; Cheeseman and Stutz, 1996; Sahami et al., 1998] are commonly used supervised learning algorithms; each algorithm captures different information. SVM is a deterministic binary classifier that generates a boundary that splits labeled data based on respective class labels. The boundary acts as a classifier to classify new unseen data. RF is an ensemble of classifiers that constructs a set of decision trees at training time. The new data is assigned to a class that is the mode of classes of individual trees. NB and its variant NBM (Naïve Bayes Multinomial) are probabilistic generative models that take into account class distributions in the process of training the classifier. Studies reveal that, with appropriate preprocessing, NBM is a popular, competitive method compared to other advanced learning algorithms such as SVM and RF [Rish, 2001; Rich and Alexandru, 2006; Jason et al., 2003; Huang et al., 2003], especially for text categorization problems, which are based on word frequencies.

Because biological sequence classification is similar to text categorization in terms of feature representation (frequencies) and format of the data (set of alphabets), and given that NBM does not have any parameters that need to be tuned, NBM was used as the base classifier for all the experiments conducted in this work.

Naïve Bayes: NB makes the assumption that features are independent given the class. Let $F = \{f_1, f_2, \dots, f_n\}$ be the set of n features used to represent a data sample and $C = \{c_1, c_2, \dots, c_k\}$ be a set of k classes of the data. Then, an instance of a data sample, X , can be represented using the set of n features as x_1, x_2, \dots, x_n , being values of the features f_1, f_2, \dots, f_n . Based on the independence assumption, the probability of an instance given a class, $P(X|c_i)$, referred to as likelihood, for $(1 \leq i \leq k)$ can be written as the product of probabilities of all features given the class:

$$P(X|c_i) = \prod_{j=1}^n P(x_j|c_i) \quad (2.1)$$

The testing phase in the NB algorithm involves the computation of the posterior probability (probability of each class given the features of the test sample), which is computed

using Bayes Theorem as follows:

$$P(c_i|X) = \frac{P(c_i) \times P(X|c_i)}{P(X)} \quad (2.2)$$

The label for a new data sample (y) is assigned based on the class that has maximum posterior probability:

$$c_{new} = \underset{i \in \{1 \dots k\}}{\operatorname{argmax}} P(c_i) \times \prod_{j=1}^n P(x_j|c_i) \quad (2.3)$$

Learning/training a classifier reduces to learning $P(c_k)$ and $P(x_n|c_k)$ for all n features and k classes. NBM is a version of NB that uses the multinomial distribution to estimate likelihood, which is specifically used when features are discrete (specifically, counts) but not binary.

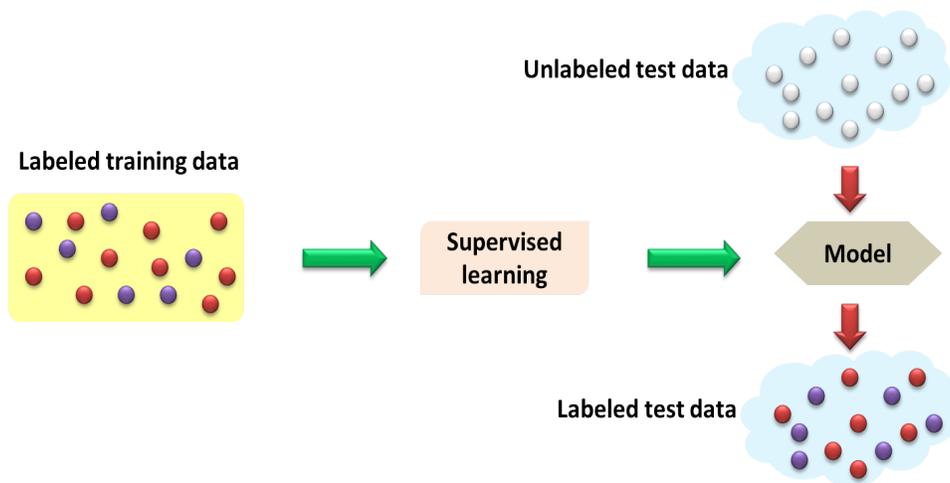


Figure 2.5: *Abstract working of the supervised learning scenario.*

Semi-supervised Learning

SSL uses knowledge from labeled and unlabeled data to learn models that can be used to predict unseen test data. SSL is effective when very little labeled data and large amounts of unlabeled data are available. Figure 2.6 shows an abstract working of SSL. Self-training

(ST) [Yarowsky, 1995] and co-training (CT) [Blum and Mitchell, 1998] are two prominent iterative-based approaches under the SSL framework that were used in this work. Algorithmic details of CT and ST are provided below.

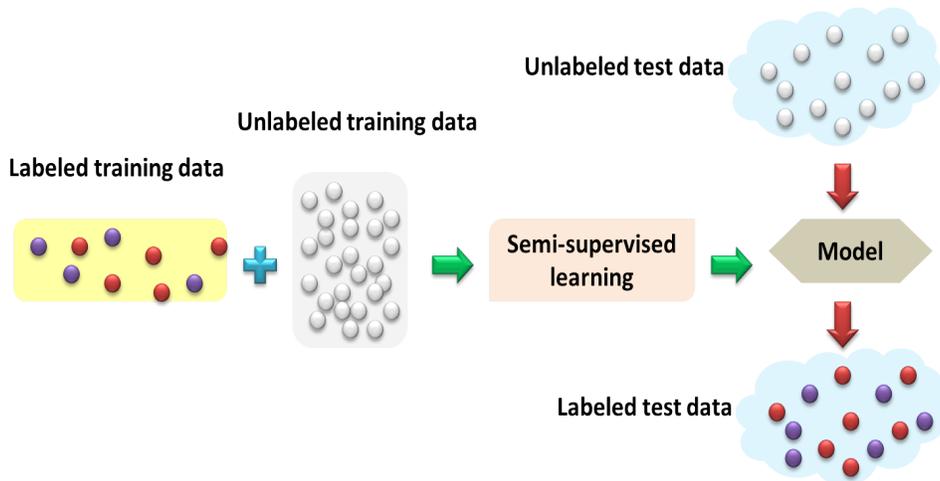


Figure 2.6: *Abstract working of the semi-supervised learning scenario.*

Self-training: ST [Yarowsky, 1995], also known as self-teaching or bootstrapping, is an iterative SSL algorithm. Input to the ST algorithm is a set of labeled instances, L , unlabeled instances, U , and test instances, T . The objective is to use L and U to iteratively generate a classifier, C , that can accurately classify new unseen test T . Let $iMax$ be the maximum number of iterations. The following pseudo-code describes the working of ST algorithm:

At each iteration, i , a base classifier, C_i is learned from L_i (initially, $L_i = L$), and C_i is then used to predict U_i (initially, $U_i = U$). Predicted instances, U_i^p , are sorted based on prediction probabilities and the most confident predictions (i.e., highest prediction probabilities), $Top(U_i^p)$, are added to the labeled set for the next iteration (L_{i+1}), such that the original class distribution is preserved. Therefore, the amount of labeled data is increased at the end of each iteration, while the amount of unlabeled data is decreased ($L_{i+1} > L_i$ and $U_{i+1} < U_i$). This process is repeated for a certain number of iterations ($iMax$) or until completion of all unlabeled instances. At each iteration, the classifier is expected to improve by including the most confident unlabeled predictions. The final base classifier C_{iMax} is then

Input: Labeled data, L ; unlabeled data, U ;
 Output: Self-training classifier - C_{ST} ;
 $i = 1$;
 $L_i = L$;
 $U_i = U$;
while ($i \leq iMax$) and ($size(U_i) > 0$) **do**
 Learn C_i from L_i ;
 $U_i^p \leftarrow$ Predictions of U_i using C_i ;
 $Top(U_i^p)$: Top predictions from U_i^p ;
 $U_{i+1} = U_i - Top(U_i^p)$;
 $L_{i+1} = L_i + Top(U_i^p)$;
 $i = i + 1$;
end
 return C_{iMax} ;

Algorithm 1: Pseudo-code for learning a self-training classifier (C_{ST}).

used to predict the test data, T .

Co-training: CT [Blum and Mitchell, 1998], another iterative SSL algorithm, is a two-view learning process. In CT, the initial feature set is split into two independent and sufficient sets of features or views [Nigam and Ghani, 2000].

- Independence assumption: Given the class, features in one view should be conditionally independent of features in the other view.
- Sufficiency assumption: Each view, by itself, should be sufficient for classification given sufficient amounts of labeled data.

Labeled, unlabeled, and test instances are then represented using the two views of features. Let L^1 , U^1 , and T^1 be labeled, unlabeled, and test data represented in one view, and L^2 , U^2 , and T^2 in the other view, respectively. Let $iMax$ be the maximum number of iterations. The two views are generated by splitting the features, but not the instances. Each view contains a different feature representation of instances corresponding to the same set of sequences. The following pseudo-code describes the working of CT algorithm:

At each iteration i , two base classifiers C_i^1 and C_i^2 are learned from L_i^1 and L_i^2 and used to predict U_i^1 and U_i^2 . Predicted instances, U_i^{1p} and U_i^{2p} , are sorted and, based on prediction

Input: Labeled data, L ; unlabeled data, U ;
 Output: Co-training classifier, C_{CT} ;
 $i = 1$;
 Represent L as L^1 and L^2 ;
 Represent U as U^1 and U^2 ;
 $L_i^1 = L^1$; $L_i^2 = L^2$;
 $U_i^1 = U^1$; $U_i^2 = U^2$;
while ($i \leq iMax$) and ($size(U) > 0$) **do**
 Learn C_i^1 from L_i^1 ;
 Learn C_i^2 from L_i^2 ;
 $U_i^{1p} \leftarrow$ Predictions of U_i^1 using C_i^1 ;
 $U_i^{2p} \leftarrow$ Predictions of U_i^2 using C_i^2 ;
 $Top(U_i^{1p})$: Top predictions from U_i^{1p} ;
 $Top(U_i^{2p})$: Top predictions from U_i^{2p} ;
 $U_{i+1}^1 = U_i^1 - Top(U_i^{1p})$;
 $L_{i+1}^2 = L_i^2 + Top(U_i^{1p})$;
 $U_i^2 = U_i^2 - Top(U_i^{2p})$;
 $L_i^1 = L_i^1 + Top(U_i^{2p})$;
 $i = i + 1$;
end
 $C_{CT} = C_{iMax}^1 \& C_{iMax}^2$;
 return C_{CT} ;

Algorithm 2: Pseudo-code for learning a co-training classifier (C_{CT}).

probabilities, confident unlabeled examples of one view are added to the labeled set of the opposite view for the next iteration (best predictions of U_i^1 to L_{i+1}^2 and best predictions of U_i^2 to L_{i+1}^1), such that original class distribution is preserved. Intuitively, instances best predicted by one classifier in one view, may not be well predicted by the classifier in the other view. Transferring information to the other classifier in this way can increase the performance of both classifiers. This process is repeated for a certain number of iterations ($iMax$) or until the completion of all unlabeled instances. Test instances (T^1 and T^2) are predicted based on mutual agreement between the two final base classifiers (C_{iMax}^1 and C_{iMax}^2).

We should note that the performance of CT is heavily dependent on the views that are used to represent the data. CT is expected to give best results when the two views are sufficient and independent.

Domain Adaptation

When minimal or no labeled data is available, but ample amounts of unlabeled data and a large corpus of labeled data is accessible for a similar problem, an alternative to SSL is to use a supervised classifier, which is trained on the labeled data from a similar problem, called the *source* domain, to label data from the problem of interest, called the *target* domain. However, although this classifier would be accurate on the data from the source domain, its classification accuracy would typically decrease for data from the target domain. A better alternative is to use a classifier in a domain adaptation setting and utilize advantageously the large volume of unlabeled data from the target domain and large volume of labeled data from a related source domain, as well as labeled data from the target domain. Figure 2.7 shows an abstract working of the domain adaptation algorithm.

The primary step in the domain adaptation algorithm is to filter out source specific features and represent source and target domain data using only selected target features. Let sL be total data from the source domain (used as labeled), tL be labeled data from

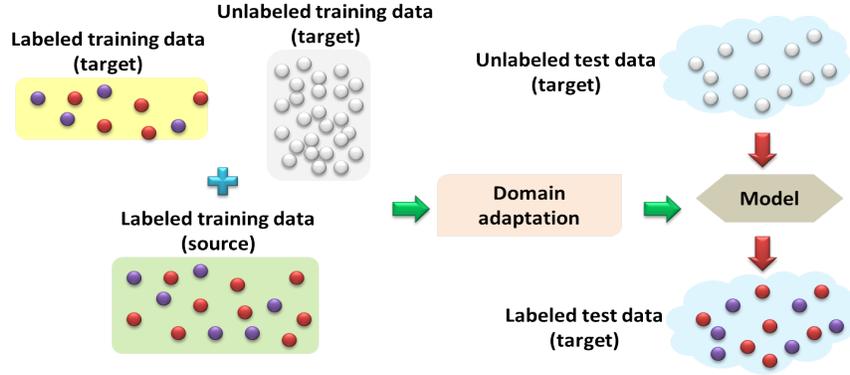


Figure 2.7: *Abstract working of the domain adaptation scenario.*

the target domain, tU be unlabeled data from the target domain, and tT be test data from the target domain, represented using selected target features. The following pseudo-code describes the process of using knowledge from source and target data to learn a domain adaptation classifier:

Similar to ST, during the first iteration, the target unlabeled data (tU) is predicted using only the source and target labeled datasets (sL and tL). Then, a number of instances, proportional to the class prior, that have highest probabilities for their corresponding class ($Top(tU_i^p)$), are selected from the unlabeled dataset at each iteration, and are used as labeled instances in subsequent iterations. Remaining instances in the unlabeled dataset are used with “soft-labels” (i.e., with weight proportional to their probability distributions) in subsequent iterations, unless they “became labeled” through ST. The process is iterated until convergence, i.e., until labels for instances in the unlabeled dataset no longer change. The resulting classifier is used to predict target test data (tT).

2.2.2 Feature Representation and Dimensionality Reduction

For any learning algorithm, the time required to train a classifier or the time the classifier requires to classify test data is highly dependent on the total number of features used to represent the respective data. Reducing the dimensionality of the feature space can benefit the learning algorithm. When sufficient amount of labeled data is available, feature selection

Input: Source labeled, sL ; target labeled, tL ; target unlabeled, tU ;
 Output: Domain adaptation classifier - C_{DA} ;
 $i = 1$;
 $tL_i = tL$;
 $tU_i = tU$;
 Learn C_i from tL_i and sL ;
 $tU_i^p \leftarrow$ Predictions of tU_i using C_i ;
 $Top(tU_i^p)$: Top predictions from tU_i^p ;
 $tU_{i+1} = tU_i - Top(tU_i^p)$;
 $tL_{i+1} = tL_i + Top(tU_i^p)$;
repeat
 $i = i + 1$;
 Learn C_i from tL_i , tU_i^p and sL ;
 $tU_i^p \leftarrow$ Predictions of tU_i using C_i ;
 $Top(tU_i^p)$: Top predictions from tU_i^p ;
 $tU_{i+1} = tU_i - Top(tU_i^p)$;
 $tL_{i+1} = tL_i + Top(tU_i^p)$;
until ($tU_i^p == tU_{i-1}^p$);
 return C_{iMax} ;

Algorithm 3: Pseudo-code for learning a domain adaptation classifier (C_{DA}).

techniques select top informative features that can be used to represent the data. This section presents details of the traditional sliding window-based approach used to generate sequential features, k -mers, to represent biological sequences. This section also provides details of feature selection (Section 2.2.2) and feature hashing (Section 2.2.2) techniques, which are primarily used to reduce dimensionality of input feature space.

Sliding Window-based Approach

In order to generate sequential features using the sliding window-based approach, a window of a particular size, k , is traversed across the sequence and the fragment of the sequence within the window is captured. All possible unique subsequences or fragments (referred to as k -mers) are used as features to represent biological sequences. As features of variable length are more informative than features of fixed length, the size of the window, k , is varied to capture variable length features. However, variable length k -mers result into a

high dimensional feature set, leading to increased computational complexity and decreased classification accuracy.

Therefore, feature selection techniques are commonly used to reduce dimensionality of the feature space, while retaining most informative features, as described in Section 2.2.2.

Feature Selection

A majority of feature selection techniques are based on the class distribution of instances containing a feature (a.k.a., document frequency, or df , in text classification) and do not take into account frequency of occurrence of a feature value within an instance (a.k.a., term frequency, or tf , in text classification). [Largeron et al. \[2011\]](#) introduced *Entropy-based Category Coverage Difference* (ECCD) that utilizes tf instead of df to compute entropy-based dependencies between features and classes.

Let M_j represent the number of instances belonging to class c_j . $\text{Max}(M_{1..c})$ returns the number of instances belonging to the majority class. The multiplication ratio r_j of a particular class c_j can be computed as follows:

$$r_j = \frac{\text{Max}(M_{1..c})}{M_j} \quad (2.4)$$

Let t_i^j be the number of occurrences of a particular feature f_i in class c_j (tf for a set of instances belonging to the same class). To account for imbalanced data, ECCD was modified by multiplying t_i^j with a multiplication ratio r_j corresponding to c_j [[Tangirala and Caragea, 2014b](#)], resulting in n_i^j :

$$n_i^j = t_i^j \times r_j \quad (2.5)$$

For a set of d documents, frequency fr_i^j of a particular feature f_i in category c_j is given by

$$fr_i^j = \frac{n_i^j}{\sum_d n_i^d} \quad (2.6)$$

Shannon entropy $SE(f_i)$ of feature f_i [Shannon, 1948] can be computed by:

$$SE(f_i) = - \sum_{j=1}^c (fr_i^j) \times (\log_2 (fr_i^j)) \quad (2.7)$$

Then $ECCD(f_i, c_j)$ is defined as

$$ECCD(f_i, c_j) = (P(f_i|c_j) - P(f_i|\bar{c}_j)) \times \frac{E_{max} - SE(f_i)}{E_{max}} \quad (2.8)$$

where $P(f_i|c_j)$ is the probability of feature f_i occurring in instances belonging to class c_j and $P(f_i|\bar{c}_j)$ is the probability of feature f_i occurring in instances belonging to all classes other than c_j . E_{max} is the maximum Shannon entropy of all features (Equation 2.9).

$$E_{max} = \underset{i=1..n}{argMax}(SE(f_i)) \quad (2.9)$$

In this work, ECCD was used to compute feature-class dependency values. All the features were then sorted based on ECCD scores and most informative features (top features) were selected. When sorting features, the class with the highest ECCD score is considered to obtain a final score of the feature.

Feature Hashing

When the amount of available labeled data is not sufficient (semi-supervised and domain adaptation learning scenarios), feature selection techniques, which rely on labeled data, may not accurately capture feature class dependency scores. Therefore, unsupervised dimensionality reduction techniques that reduce dimensionality of the input feature space without using class labels are essential. Feature hashing [Weinberger et al., 2009; Shi et al., 2009; Forman and Kirshenbaum, 2008] is an unsupervised dimensionality reduction technique that

hashes a high-dimensional input feature space into low-dimensional feature vectors of pre-defined size b . The feature hashing technique by Caragea et al. [2012] was used in this work. This section, presents background details and working of the feature hashing technique used to reduce the dimensionality of k -mers [Caragea et al., 2012].

Let K denote the total set of k -mers of desired length and h and ξ be two hash functions such that $h : K \rightarrow \{0, \dots, b - 1\}$ and $\xi : K \rightarrow \{\pm 1\}$, respectively. Each k -mer corresponding to a biological sequence is directly mapped to one of the b bins using the hash function, h . The hash key obtained for each k -mer is a number between 0 and $b-1$ that refers to the index of the target bin. The frequency count of the current k -mer in the current sequence is then added to the total count of the resulting bin. Therefore, entry i in the vector records frequency counts of all k -mers that generated the hash key, i .

The function h can be any hash function; this work used the `hashCode()` function of the Java String class, which uses a product sum algorithm over the complete string. If s is the target string of length n , then the Java `hashCode()` for s is given by Equation 2.10, where sum (\sum) refers to the Java 32-bit `int` addition and $s[i]$ is the i^{th} term in the string s :

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i} \quad (2.10)$$

The hash function ξ , which returns either $+1$ or -1 , determines whether to increment or decrement the target bin. Use of this function can reduce the bias (several strings with the same hash key) created by the hash function (see [Weinberger et al., 2009] for more details).

Feature hashing allows multiple k -mers to be hashed to a single bin, i , based on the resulting hash key. Therefore, a very small value of b can be problematic as the number of collisions increases by a large extent, resulting in significant loss of information [Caragea et al., 2012].

Chapter 3

Related Work

This chapter reviews previous work on the applications of Burrows Wheeler Transform and community detection approaches in bioinformatics in Sections 3.1 and 3.2, respectively. Section 3.3 reviews previous work that used features constructed using the sliding window-based approach and Section 3.4 details various feature selection techniques. Finally, Section 3.5 outlines approaches designed to construct or identify patterns and motifs from biological sequences.

3.1 Burrows Wheeler Transform in Bioinformatics

BWT was first introduced by [Burrows and Wheeler \[1994\]](#) to address the problem of data compression. The ability of BWT to efficiently identify multiple occurrences of a particular subsequence within a sequence generated significant interest in this approach, especially in the field of bioinformatics. Several applications on BWT have been developed for various biological problems.

[Ferragina and Manzini \[2000\]](#) proposed an approach called FM-index that uses the BWT along with the suffix array data structure to efficiently find the number of pattern occurrences within a text. In addition to determining the count, the FM-index also identifies the

location of all patterns in the original sequence. The authors proposed an algorithm whose running time and storage space are sub-linear with respect to size of the data. BWT has been used to facilitate alignment of short oligonucleotides. For example, Li et al. [2009] developed SOAP2, a tool that replaced the original seed strategy of SOAP [Li et al., 2008b] with BWT indexing, thereby reducing memory usage and increasing alignment speed.

Li et al. [2009] also aligned short reads to a longer sequence (specifically genome) using BWT. The approach [Li et al., 2009] used the backward search with BWT and performed top-down traversal on the prefix trie of the genome. The authors computed the number of exact matches of a sequence of length m in $O(m)$ time independent of the genome size. The authors compared the performance of their approach (called BWA) with MAQ [Li et al., 2008a], SOAP2 [Li et al., 2009], and Bowtie [Langmead et al., 2009]; in most cases, BWA outperformed MAQ, SOAP2 and Bowtie in terms of computational time. Kulekci et al. [2012] developed a BWT-based approach for finding maximal repeats in a given input sequence. The approach used Wavelet Trees along with BWT and suffix arrays to initially identify candidate repeats. The authors also sought to determine if candidate repeats could be extended to the left of the sequence. The approach [Kulekci et al., 2012] improved the efficiency of repeat finding for a large sequence compared to state-of-the-art suffix array-based implementation [Becher et al., 2009]. Other approaches such as [Danek et al., 2012; Rafal and Andrzej, 2010] also use BWT to find repeats in a set of sequences. To the best of the author's knowledge, BWT has never been used to generate features for biological sequence classification.

3.2 Community Detection in Bioinformatics

In modern sciences, many complex systems can be represented using a network, with nodes as the elementary components of the system and links as the relationships between components. A community is a subnetwork with nodes that are strongly connected compared to

other network nodes. Identification of communities in a complex system can uncover structural, topographical, and/or relational properties of the system. Several approaches have recently been proposed to identify communities in networks. [Girvan and Newman \[2002\]](#) and [Newman and Girvan \[2004\]](#) proposed a hierarchical divisive algorithm, which is believed to be the first algorithm of modern day community detection algorithms. The algorithm iteratively removed edges between the nodes based on their “betweenness,” thereby defining the number of shortest paths between nodes that pass through the edge. The process of removing edges stops when the modularity of the partition reaches the maximum. The authors used a modularity function (Newman-Girvan modularity) that defined the quality of the partition by comparing it with a null model (random graph). [Clauset et al. \[2004\]](#) defined a fast approach that begins with a set of isolated nodes in which edges are iteratively added based on modularity (Newman-Girvan modularity) gain. Some techniques use exhaustive optimization in order to better estimate the final maximum modularity at the expense of computational cost [[Guimera, 2004](#); [Massen and Doye, 2005](#); [Medus et al., 2005](#); [Guimera and Amaral, 2005](#)]. [Blondel et al. \[2008\]](#) defined a fast multi-step technique that identified near optimal communities (locally optimal). At each iteration, the authors replaced communities with super-nodes, thereby producing a smaller network for the next iteration. This process was repeated until modularity no longer increased. Several other techniques have been proposed to identify communities within large complex networks [[Raghavan et al., 2007](#); [Rosvall and Bergstrom, 2008](#); [Radicchi et al., 2004](#); [Donetti and Muñoz, 2005](#)]. The community detection algorithm proposed by [Blondel et al. \[2008\]](#) was used in this work to identify motifs or groups of related sequences and use the sequences belonging to motifs to classify biological sequences.

In bioinformatics, community detection has been used primarily in the context of protein-protein interaction networks and prediction of functional families [[Dongfang and Xiaolong, 2013](#); [Mahmoud et al., 2014](#); [Mallek et al., 2015](#); [van Laarhoven and Marchiori, 2012](#)]. [Jia et al. \[2013\]](#) used community detection to identify transcription factor binding sites in a

small set of nucleotide sequences in an approach referred to as TFBSGroup. To the best of the author’s knowledge, community detection algorithms have not been used to construct sequential features for biological sequence classification problems in a machine learning framework.

3.3 Sliding Window Approach in Bioinformatics

K -mers generated using the sliding window-based approach are used when biologically informative sets of motifs are not available. K -mers have been used to represent sequences for several sequence classification problems. Spectrum and mismatch kernels, used primarily for protein sequence classification, require sequences to be represented using k -mers [Leslie et al., 2004, 2002]. Caragea et al. [2012] used feature hashing with k -mers in order to reduce dimensionality for protein sequence classification. Sun et al. [2013] distinguished protein-coding and non-coding transcripts using sequence composition models (based on k -mers). K -mers have also been used in genome and transcriptome assembly, error correction of reads, and metagenomic sequencing [Melsted and Pritchard, 2011].

3.4 Feature Selection

In sequence-based classification problems and more importantly when we are working with k -mers, several features might act as outliers. In such cases, using all the features can mislead the classifier, thereby affecting the performance of the classifier. Feature selection addresses the problem of removing features that are not sufficiently informative by computing, for example, mutual information between each feature and the class variable. Feature selection can be essential in improving the performance of the classifier, in addition to reducing the dimensionality of the input feature space, which affects efficiency. Various feature selection techniques have been proposed [Ng et al., 1997; Wiener et al., 1995; Battiti, 1994].

Because of the dependency between adjacent characters of a sequence, many Markov

models have been developed to select features from biological sequences. [Salzberg et al. \[1998\]](#) used interpolation between different orders of Markov models, known as interpolated Markov model (IMM), and a filter (χ^2 test) to select a subset of features. [Saeys et al. \[2007\]](#) used the Markov blanket multivariate approach (MBF) on top of a combination of different measures of coding potential prediction in order to retain informative features. [Chuzhanova et al. \[1998\]](#) combined a genetic algorithm with a Gamma test to obtain scores for feature subsets. The optimal subset was then selected based on the resulting scores. [Zavaljevski et al. \[2002\]](#) used selective kernel scaling for SVMs to compute weights of the features. Features with low weights were subsequently ignored. [Degroeve et al. \[2002\]](#) addressed the problem of splice site prediction through feature selection, using a sequential backward method and an embedded evaluation criterion based on SVM. However, these approaches are computationally expensive and require a lot of time to select informative sequential features.

Alternatively, when sequences are represented using frequency distribution of various subsequences (features), feature selection techniques that estimate feature-class dependencies using frequency distribution of instances containing a feature (a.k.a., document frequency, or df , in text classification) are used [[Ng et al., 1997](#); [Galavotti et al., 2000](#); [Caropreso et al., 2001](#); [Hanchuan et al., 2005](#)]. However, these approaches do not take into account the frequency of occurrence of a feature value within an instance (a.k.a., term frequency, or tf , in text classification). [Largeron et al. \[2011\]](#) introduced *Entropy-based Category Coverage Difference* (ECCD) that makes use of tf in place of df to compute the entropy based dependency between features and classes.

3.5 Sequential Pattern Mining Algorithms

Motif discovery has recently attracted researchers, resulting in development of several approaches and tools to identify motifs in biological sequences [[Kjetil Sandve and Drabls, 2006](#);

Das and Dai, 2007; Zambelli et al., 2012]. Some approaches use probabilistic/statistical models to identify motifs, primarily represented in the form of position-specific scoring matrices (PSSM) [Bucher, 1990], such as Multiple EM for Motif Elicitation (MEME) [Bailey and Elkan, 1995], NMica [Down and Hubbard, 2005], AlignACE [Roth et al., 1998], MDscan [Liu et al., 2002], Yeast Motif Finder (YMF) [Sinha and Tompa, 2003], Gibbs Sampler [Lawrence et al., 1993], PROJECTIONS [Buhler and Tompa, 2001], DRIMust [Leibovich et al., 2013] and CRMD [Li et al., 2010]. Although these approaches can have a fast run time, attaining global optimum when identifying motifs with these approaches is sometimes difficult, especially when the motif length is small. Other approaches such as MITRA-count [Eskin and Pevzner, 2002], TFBSGroup [Jia et al., 2013], WEEDER [Pavesi et al., 2001, 2004], *L. SPELLER* [Sagot, 1998], Voting approach [Xu et al., 2013], WINNOWER [Pevzner and Sze, 2000], Stemming [Kuksa and Pavlovic, 2010], RecMotif [Sun et al., 2010], and sMCL-WMR [Boucher and King, 2010] typically search for (l, d) motifs based on the consensus model using various heuristic methods.

Several sequential mining algorithms have also been proposed to generate patterns and motifs from biological sequences. Exarchos et al. [2006, 2008] and Fotiadis et al. [2007] used sequential pattern mining (SPM) [Agrawal and Srikant, 1995] to generate patterns for protein sequence classification. Liao and Chen [2013, 2014] used a recursive depth-first approach to identify sequential patterns with and without gaps. Exarchos et al. [2008] used cSpade [Zaki, 2001], an efficient SPM algorithm, to identify patterns for protein fold recognition. Wang et al. [2004] introduced a tree-based SPM approach that first identifies small patterns, known as segments, and then searches for long patterns that contain multiple segments. Vens et al. [2011] used a combination of directed acyclic graphs and segment trees to identify patterns from sequences belonging to the positive class. Several other approaches have also successfully identified or generated patterns from sequential data [Zhu et al., 2007; Kang et al., 2007; He et al., 2007; Wu et al., 2013; Chen and Liu, 2013; Liao and Chen, 2012].

Although these approaches effectively identify motifs (e.g., transcription factor binding sites) within a small set of sequences, they cannot be used to construct sequential features for machine learning algorithms (because of the large data set size). Alternatively, in this dissertation, scalable approaches (in terms of dataset size) to construct sequential features, specifically for biological sequence classification problems were proposed.

Chapter 4

Proposed Feature Construction

Approaches

As discussed in Section 2.2.2 (sliding window-based approach), for large datasets, the dimensionality of the set of k -mers increases exponentially with k , thereby increasing the learning and classification time of the learning algorithms. Therefore, feature selection techniques are commonly used to reduce dimensionality of the input feature space, while retaining a majority of the informative features. However, as feature selection techniques use the available labeled data in the process of selecting informative features, such techniques may not be accurate in semi-supervised and domain adaptation algorithms, because the amount of available label data is small.

This work, however, proposes three novel unsupervised (class labels are not required) approaches to generate low-dimensional informative sequential features (a subset of variable length k -mers) to represent biological sequences. Section 4.1 details the BWT-based approach to generate a reduced set of sequential features. Section 4.2 describes the process of using the CDA-based approach to generate a reduced set of k -mers, and Section 4.3 describes the process of combining the BWT-based approach with the CDA-based approach to obtain BWT-based features with certain mismatches.

4.1 Burrows Wheeler Transform Approach

This section describes the process of using BWT to generate features from biological sequence data, specifically nucleotide and protein sequences. Section 4.1.1 explains the process of generating BWT of a given sequence, while Section 4.1.2 describes the process of using BWT to generate sequential features.

4.1.1 Burrows Wheeler Transform Preliminaries

BWT is a context-dependent permutation of the input sequence, such that characters adjoining similar suffixes (a sequence of characters or patterns occurring multiple times) are grouped together. In what follows, a sequence S is denoted with a set of consecutive characters $c_1c_2c_3\dots c_{n-1}c_n$. BWT of S can be obtained as described below.

Generating BWT of a sequence: If $c_1c_2c_3\dots c_{n-1}c_n$ is a sequence, then $c_nc_1c_2c_3\dots c_{n-1}$ is called a *rotation* R of the sequence, obtained by removing the last character of the sequence and appending it at the beginning. A sequence of length n can have a maximum of n *rotations*, denoted by an array of *rotations*, $R[1..n]$. The *rotations* $R[1..n]$ are sorted alpha-numerically and the last column of the sorted *rotations* ($sort(R[1..n])$) is the BWT of the input sequence.

Figure 4.1 (a) shows the process of generating BWT for an example sequence *ragtlagtgytlag*. A special character \$ was appended to mark the end of the sequence. The last column of the sorted *rotations*, *gllraattt\$gygg*, is the BWT of the input sequence.

4.1.2 Feature Construction based on Burrows Wheeler Transform

Characters in $BWT(S)$, the last column of sorted *rotations*, $sort(R[1..n])$, are grouped based on similarities among the prefixes of corresponding *rotations*; the array $R[1..n]$ is sorted alpha-numerically. Let i denote the index of a *rotation* in $sort(R[1..n])$, where $1 \leq i \leq n$. As shown in Figure 4.1, for $i = 2$ and 3 , the character l has two occurrences in the

last column (grouped together as rotations at indices 2 and 3) that have the same prefix ag . The prefix of a *rotation*, $sort(R[i])$, is the suffix to the corresponding last character in the original sequence S because each *rotation* is a transformation of the original sequence, in which the last character is removed and appended at the beginning. Therefore, the characters in $BWT(S)$ are grouped based on similarities among corresponding suffixes in the original sequence, S . This work exploits this property of BWT to group prefixes based on lexicographically similar suffixes in order to generate variable length features that occur multiple times in a given sequence.

Consecutive occurrences of a character x in $BWT(S)$, referred to as a *repetition*, were sought out in order to generate features (subsequences). Let $start$ and end be the starting and ending indices of a *repetition* in $BWT(S)$. *Rotations* at indices i (where $start \leq i \leq end$) were selected from $sort(R[1..n])$, and a common prefix (γ) was sought among the selected *rotations*. The common prefix, γ , was then appended to the repeated character, x , and $x + \gamma$ was returned as a feature. A *repetition* of length r is associated with a corresponding feature occurring at least r times in the original sequence ($r = end - start + 1$). In this work, features and/or subsequences that occurred at least twice in the original sequence, *i.e.*, $r \geq 2$, were selected.

Features returned by the BWT-based approach were of variable length, specifically $|\gamma| + 1$. From the set of all features returned by the approach, the features that match the desired length were selected. The resulting set of features was referred to as ***b-mers***.

Although features that occurred at least twice in at least one sequence were returned, not all subsequences that occurred at least twice were returned as features because, in addition to frequency, the proposed approach also considered other properties based on possible suffixes and lengths of each feature. The properties are described in what follows. These properties are verified implicitly when using the procedure described above (BWT-based approach), so they need not be checked explicitly. The following notations were used to make the presentation precise:

For a sequence S , let α be a subsequence of S . For a sequence, a subsequence on the left side of the sequence was referred to as a left segment, denoted by *leftSeg*, and a subsequence on the right side of the sequence was referred to as a right segment, denoted by *rightSeg*. For example, for the sequence “tgct,” “t,” “tg,” “tgc,” etc., can be *leftSeg*, while “t,” “ct,” “gct,” etc., can be *rightSeg*. Let $|\alpha|$ be the length of the subsequence α . Features generated with the BWT-based approach should satisfy all of the following properties:

1. A feature occurs at least two times in the original sequence, S .
2. If different subsequences with identical *leftSeg* have same frequency of occurrences, then the BWT-based approach returns only the subsequence of maximum length that satisfies all other properties, as a feature.
3. If a subsequence α of S has a *leftSeg* with frequency of occurrence in S greater than the frequency of α in S , then the BWT-based approach returns that *leftSeg* as a feature if either *start* or *end* indices associated with *leftSeg* are adjacent or between the *start* and *end* indices associated with α . Otherwise, BWT returns α as a feature if all other properties are satisfied by α .
4. If α occurs multiple times in S , the BWT-based approach returns α as a feature if no other subsequence of S , β , exists, such that α and β have identical *leftSeg*. If α and β have identical *leftSeg*, BWT returns the *leftSeg* that is common to both α and β as a feature.
5. If α and β are two subsequences of S with identical *rightSeg*, preceded by two different characters, then the BWT-based approach returns α if and only if at least two rotations corresponding to α are grouped together in the sorted rotations (i.e., they are not interspread with rotations corresponding to β) and if no other subsequence(s) of S , δ , with length greater than α is associated with the same set of grouped rotations (i.e., if *rightSeg* of δ , with length $|\delta|-1$ is a common prefix of all grouped rotations). If such

subsequence (s) exists, BWT returns the subsequence of maximum length (longest of all δs) associated with grouped rotations as feature, ignoring α .

Example: Figure 4.1-(b) shows the process of generating b -mers for an example sequence. Let the minimum length of the *repetitions*, r , be 2 and the minimum length of the features be 2 (indicating that length of $\gamma \geq 1$). In $BWT(S)$, which is $gllraaat\text{\$}gyaa$, four *repetitions* can be observed: ll , aaa , ttt , and gg . Consider the first *repetition*, ll ($x = l$), with the starting index in $BWT(S)$, $start = 2$, and the ending index, $end = 3$. As $r \geq 2$ ($r = 3 - 2 + 1$), *rotations* at indices 2 and 3 were selected from $sort(R[1..n])$. The selected *rotations* had a common prefix, $\gamma = ag$. As the length of γ was greater than or equal to 1, γ was appended to x and lag was returned as a feature. Similarly, for repetition aaa , ag was returned as a feature. However, for *repetitions* ttt and gg , length of the common prefix, γ , was zero, which is less than the desired length. Therefore, no features associated with these *repetitions* were returned. As a result, b -mers associated with sequence $S = ragt\text{\$}lagtgyt\text{\$}lag$ were lag and ag . In addition to lag and ag , several other subsequences occurred at least twice in the original sequence (agt , gt , tl , la) but were not returned by the proposed approach because the BWT-based approach also considered possible suffixes and lengths of each features as follows:

- agt : agt had a *leftSeg*, ag with frequency greater than the frequency of agt , and the *start* and *end* indices of ag (5 and 7) were adjacent to indices corresponding to agt (6 and 7). Therefore, based on the third property of the BWT-based approach, *leftSeg*, ag was returned as a feature, instead of agt .
- gt : In addition to gt , subsequence yt that also appeared in the sequence, had identical *rightSeg*, t , preceded by different characters. Therefore, based on the fifth property of the BWT-based approach, at least two rotations of gt should not be interleaved by rotations corresponding to yt . However, the index of the rotation corresponding to yt , 13, was in between the two indices corresponding to gt , 12 and 14. Therefore, the

BWT-based approach did not return gt as a feature.

- tl : Both tl and tg had identical $leftSeg$, t . Therefore, based on the fourth property of the BWT-based approach, $leftSeg$ should be returned as the feature. However, the length of the common $leftSeg$ was 1, and did not satisfy the minimum length constraint (minimum length = 2).
- la : lag had the same $leftSeg$, la as la ; the frequency of occurrence of lag and la was 2. Therefore, based on the second property of the BWT-based approach, lag was returned as a feature, ignoring la because lag satisfied all other properties.

In conclusion, b -mers can be considered as a reduced set of k -mers, selecting k -mers that occur at least twice in at least one sequence, in addition to satisfying other properties based on length, frequency, and suffix information.

4.2 Community Detection Approach

4.2.1 Community Detection Preliminaries

Network analysis has generated increased attention among researchers interested in identifying hidden structural and relational properties within a large complex system. Similar to a graph, a network is comprised of a set of V nodes $\{n_1, n_2, \dots, n_V\}$ and a set of E edges $\{(n_i, n_j) \mid 1 \leq i \neq j \leq V\}$. Many complex systems can be represented using a network, with nodes as elementary components of the system and the relationship between components as links.

A community (as shown in Figure 4.2) is a subnetwork whose nodes are highly connected with each other, when compared to other nodes. A community within a network reflects a group of closely associated nodes. Many methods have been developed to identify communities among a network [Harenberg et al., 2014; Fortunato and Lancichinetti, 2009;

Fortunato, 2010]. In this work, a modularity gain-based, multi-step technique was used to identify communities [Blondel et al., 2008].

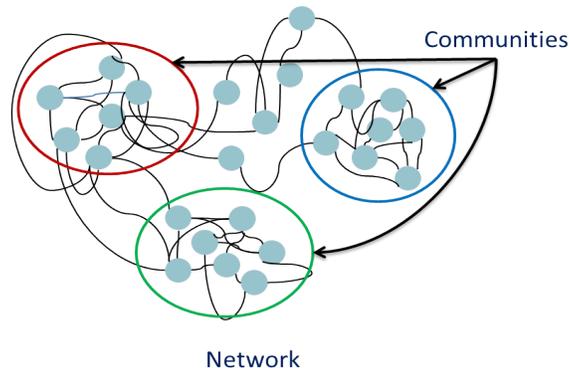


Figure 4.2: *A network and its associated communities.*

Identifying Communities by Optimizing Modularity [Blondel et al., 2008]:

The algorithm proposed by Blondel et al. [2008] identifies communities by optimizing modularity. This algorithm is a fast and efficient approach to identify high modularity partitions in a large network.

Modularity, Q , measures network structure by defining network strength when divided into modules (subnetworks or communities). High modularity of a network suggests that nodes within each community are densely connected compared to other nodes. Equation (4.1) can be used to compute modularity, where A_{ij} is the weight of the edge (n_i, n_j) :

$$Modularity = Q = \frac{1}{2E} \sum_{0 \leq i, j \leq V} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(C_i, C_j) \quad (4.1)$$

For an unweighted network, the default weight of the edge was chosen to be 1; k_i is the sum of the weights of all edges corresponding to node n_i , given by $k_i = \sum_{j=1}^n A_{ij}$; C_i is the community to which n_i is assigned, $\delta(., .)$ is the *Kronecker* function (Equation 4.2), and $m = \frac{1}{2} \sum_{ij} A_{ij}$.

$$\delta_{C_i, C_j} = \begin{cases} 0 & \text{if } C_i \neq C_j \\ 1 & \text{if } C_i = C_j, \end{cases} \quad (4.2)$$

This approach [Blondel et al., 2008] is a two-phase iterative process. In the first phase, each node is assigned to a different community. Then, for each node, n_i , the algorithm computes gain in modularity, ΔQ , achieved by removing n_i from its community and placing it in the community of n_j , where n_j is a neighboring node of n_i . Node n_i is then assigned to the community of n_j for which maximum modularity gain is obtained based on the Equation 4.3.

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \quad (4.3)$$

where \sum_{in} is sum of the weights of the edges within community C_u , \sum_{tot} is the sum of weights of all edges associated with the nodes in C_u , k_i is the sum of weights of all edges from node i , $k_{i,in}$ is the sum of weights of all edges from node i to all nodes in C_u , and $m = \frac{1}{2} \sum_{ij} A_{ij}$.

In the second phase, a new network is constructed with nodes being the communities identified in the first phase. Weights of edges between the new nodes are computed as the sum of weights of edges between nodes of the corresponding two communities. Edges among nodes of the same community form self-loops in the new network. These two phases are iterated until no further improvement is observed in the modularity gain and the final set of communities is returned.

Resulting communities can be used to determine several properties of the network. Jia et al. [2013] used this community detection algorithm to identify transcription factor binding sites in nucleotide sequences. Section 4.2.2 presents details of the approach proposed by Jia et al. [2013], referred to as TFBSGroup.

4.2.2 Identifying Motifs Using Community Detection - TFBS-Group

Jia et al. [2013] proposed the approach of using a community detection algorithm to identify transcription factor binding sites (a.k.a., motifs) in a set of nucleotide sequences. A motif is a widespread pattern across various sequences with potential biological significance. A motif can be obtained by aligning a set of highly correlated subsequences that occur across various sequences (called motif instances). The motif is also referred to as the consensus of its motif instances. TFBSGroup, the approach proposed by Jia et al. [2013], attempts to identify motifs under the ZOMOPS constraint (Zero, One, or Multiple Occurrences of a Motif Per Sequence). Identified motifs have length k , and a maximum of d mismatches between motif instances and motif consensus is allowed. For a set of N sequences of maximum length L , the TFBSGroup approach works in three phases:

1. **Network construction:** The first phase involves construction of an N -partite graph. Network nodes represent all possible l -mers (subsequences of length l) of input sequences. Therefore, for a set of N sequences, each of length L , ($N * (L - l + 1)$) nodes exist.

A pair of nodes are connected by an edge only if the Hamming distance between l -mers corresponding to the two nodes is no more than x . If the maximum Hamming distance between a motif instance and motif consensus is d , the maximum number of mismatches between any two motif instances is $2d$. Therefore, x is given a maximum value of $2d$ while the network is constructed, and the minimum value of x is chosen to be d (i.e., $d \leq x \leq 2d$) to avoid spurious edges. Two nodes (l -mers) from the same sequence are not connected by an edge; therefore, a set of N sequences generates an N -partite graph/network. Figure 4.3 shows a 2-partite network for 2 sequences, with $l = 4$, $d = 1$, and $x = 2$.

2. **Community detection:** After constructing the network, all possible communities of

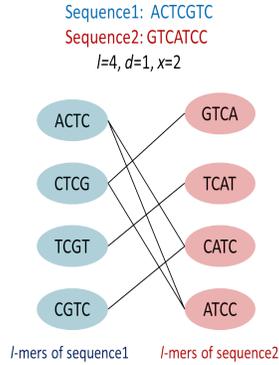


Figure 4.3: Construction of N -partite graph for two sequences with subsequence length, $l = 4$, maximum hamming distance, $d = 1$.

size at least q are identified in the N -partite network using the community detection algorithm, described in Section 4.2.1.

3. **Identifying motifs:** For each community identified in the previous phase, a motif consensus is generated by aligning all l -mers from that particular community. Each motif consensus is greedily refined towards the true motif as follows:

- (a) For each node in the community, its neighbors (i.e., set of nodes, with l -mers of the original sequence that are surrounded by l -mer corresponding to the current node) are considered. For each node in the neighbor set, if the Hamming distance between the corresponding l -mer and the motif consensus is less than or equal to d , the node is added to a new community.
- (b) The l -mers corresponding to nodes of the new community are further aligned to form a new candidate motif consensus. These two steps are iterated until the new candidate motif consensus remains unchanged.
- (c) The l -mers of nodes corresponding to the final community are shifted to the left and right because the true motif consensus and its instances may be near the final candidate motif consensus and their associated instances.

For each community, the refined motif consensus and associated motif instances are

returned together with a significance score [Pavesi et al., 2001, 2004]. The top t motifs (default value of t was chosen to be 10) are then selected based on the significance score (additional details regarding the TFBSGroup approach are presented in [Jia et al., 2013]).

Time and space complexity: Let $p(l, x)$ be the probability of two l -mers with a maximum Hamming distance of x , where $p(l, x)$ is defined as shown in Equation (4.4):

$$p(l, x) = \sum_{i=0}^x \binom{i}{l} \frac{3^i}{4} \cdot \frac{1^{(l-i)}}{4}. \quad (4.4)$$

According to Jia et al. [2013], the worst-case time complexity of TFBSGroup algorithm is $O(p(l, x)^2 \times N^4 \times L^4)$. However, since the probability term, $p(l, x)$, is bounded above by 1.0, $O(p(l, x)^2 \times N^4 \times L^4) \subseteq O(N^4 \times L^4)$; that is, it is also in the Big-O class $O(N^4 \times L^4)$.

Although TFBSGroup can successfully identify transcription factor binding sites in a small set of sequences, it cannot be applied to generate features for classification problems due to the large number of sequences involved in classification. Therefore, an approach for scaling up TFBSGroup was proposed in this work. Section 4.2.3 describes the proposed approach, referred to as the CDA-based approach, to extend TFBSGroup to generate low-dimensional informative sequential features for nucleotide sequence classification problems.

4.2.3 Feature Construction for Large Nucleotide Sequence Datasets

In order to extend TFBSGroup to generate features for sequence classification problems, TFBSGroup was invoked on randomly selected R samples, each of S sequences, from available data consisting of N sequences, where $S \ll N$, as shown in Figure 4.4. As a result, the time complexity of running TFBSGroup on a sample was reduced to

$$O(p(l, x)^2 \times S^4 \times L^4) \ll O(p(l, x)^2 \times N^4 \times L^4), \text{ as } S^4 \ll N^4.$$

For a set of R samples, the time complexity was

$$O(p(l, x)^2 \times S^4 \times L^4 \times R) \ll O(p(l, x)^2 \times N^4 \times L^4), \text{ when } (R \times S^4) \ll N^4.$$

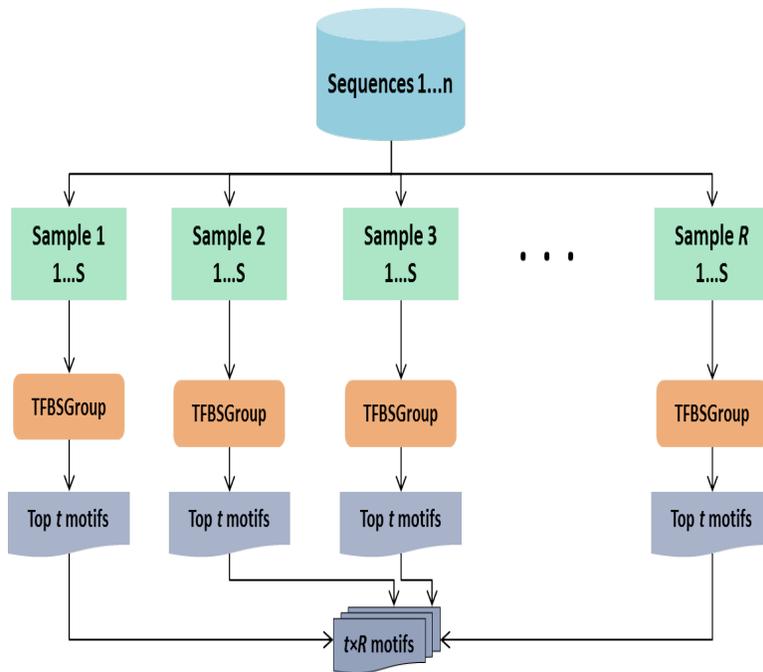


Figure 4.4: Using community detection algorithm to construct sequential features for biological sequences. Available training data containing N sequences was split into R samples, each containing S sequences. TFBSGroup was then invoked on each of the samples, and outputs of all the samples were grouped to form the final set of motifs.

In this work, R and S were chosen to achieve scalability. When generating R samples, overlap between samples was allowed, but not overlap between sequences within a sample, because the objective of the CDA-based approach was to find frequent patterns or motif instances across sequences, but not necessarily within a sequence.

TFBSGroup was invoked on each individual sample and the top t motifs from each sample were selected. All resulting motifs were merged to form the final set of motifs. As a result, the final set of motifs contained a total of $t \times R$ motifs (for a particular length of the motif, l). Figure 4.5 presents an example motif for nucleotide sequences, with motif consensus, a score that specifies motif quality, and a set of motif instances with their position ($sequence\#, startingposition$) in the set of S sequences. A motif is a set of l -mers with a maximum of $2d$ mismatches among themselves that span various sequences of a sample. The set of unique motif instances and the motif consensus were referred to as the set of **c-mers** (because they were identified based on community detection).

```
CAGTGATGCAGCAAAATCAAAGAAGTGGCTGGTATAAGTGAGAAAAGG
TGGAAATTAGAAATTTAGAGGGGGAAAATGGGAAATTGTGGAAATTG
CATATAAATGTTTAAATTTTGAGTAGAAGCTTTTCATAGCACTTTAAAA
TGTATGCCGAGAGAAAGAATTGAATATTCGAGATATTGGGTGATAACAC
GAAATGCCGATTTGAAAATGTTTGAAGTTGGAATGAAAAAGAATGT
```

```
Top 1 motif:AAATTG
score:20.883698
( 1, 4)    AAATTA
( 1, 11)   AAATTT
( 1, 26)   AAAATG
( 1, 34)   AAATTG
( 1, 43)   AAATTG
( 2, 13)   AAATTT
( 3, 17)   GAATTG
( 3, 33)   ATATTG
( 4, 15)   AAAATG
( 4, 26)   AAGTTG
```

Figure 4.5: A sample motif identified using the CDA-based approach. Motif instances, motif consensus, and the significance score of the consensus are shown.

Although this approach can be used to construct features for learning classifiers from large sets of nucleotide sequences, it cannot be directly used to construct features for learning protein sequence classifiers because the Hamming distance is not an accurate similarity measure when determining differences between short protein l -mers.

4.2.4 Feature Construction for Protein Sequence Datasets

For protein sequences, short motifs (up to length 4) carry better information than long motifs [Cheng et al., 2005; Yang et al., 2008; Caragea et al., 2012]. When the length of the motif is short (*e.g.*, $l = 1, 2$, or 3), the probability of two protein l -mers having Hamming distance less than a particular threshold, x , is high because $x \approx l$; therefore the resulting network is very dense. For long motifs (*e.g.*, $l = 6, 7$, or 8), the desired threshold x is typically smaller than l . Therefore, given the large alphabet size of protein sequences, the probability of having an edge between two nodes is very low, resulting in a very sparse network. When Hamming distance is used to construct a network of protein subsequences, the resulting network is either too sparse or too dense. In order to address this problem, a novel idea of using substitution scores in the process of constructing sequential features for protein sequences was proposed in this work. Substitution scores were computed using substitution matrices for amino acids which take into account divergence time as well as the substitution rate for each possible alignment of amino acids. Based on default parameters for BLAST, PAM30 substitution matrix [Dayhoff et al., 1978; Henikoff and Henikoff, 1992], shown in Figure 4.6, was used in this work to compute the substitution score between two protein motifs as an attempt to capture similarities between short subsequences.

Similar to Hamming distance, the substitution score for a pair of l -mers was computed based on alignment of amino acids at respective positions of the l -mers. However, contrary to using Hamming distance, when using substitution matrices, the score of alignment at a particular position is affected by the match/mismatch of the respective amino acids and by the degree of match/mismatch as captured by the substitution matrix. For example, con-

sider two pairs of 3-mers: $\{NQM, DHM\}$ and $\{PGD, RGD\}$. For pair 1, the Hamming distance is 2 and the substitution score is 14, and for pair 2, the Hamming distance is 1 and the substitution score is 10. Substitution scores were computed using PAM30 matrix [Dayhoff et al., 1978] and the scores represent similarity as opposed to distance. The higher the substitution score values, the more similar the sequences. Therefore, based on Hamming distance, l -mers of pair 2 are more similar than l -mers of pair 1. Contrarily, based on substitution scores, l -mers of pair 1 are more similar compared to pair 2. Because substitution scores capture the degree of match/mismatch, they are preferable to the Hamming distance when the objective is to identify similar protein sequences.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	6	-7	-4	-3	-6	-4	-2	-2	-7	-5	-6	-7	-5	-8	-2	0	-1	-13	-8	-2	-3	-3	-3	-17
R	-7	8	-6	-10	-8	-2	-9	-9	-2	-5	-8	0	-4	-9	-4	-3	-6	-2	-10	-8	-7	-4	-6	-17
N	-4	-6	8	2	-11	-3	-2	-3	0	-5	-7	-1	-9	-9	-6	0	-2	-8	-4	-8	6	-3	-3	-17
D	-3	-10	2	8	-14	-2	2	-3	-4	-7	-12	-4	-11	-15	-8	-4	-5	-15	-11	-8	6	1	-5	-17
C	-6	-8	-11	-14	10	-14	-14	-9	-7	-6	-15	-14	-13	-13	-8	-3	-8	-15	-4	-6	-12	-14	-9	-17
Q	-4	-2	-3	-2	-14	8	1	-7	1	-8	-5	-3	-4	-13	-3	-5	-5	-13	-12	-7	-3	6	-5	-17
E	-2	-9	-2	2	-14	1	8	-4	-5	-5	-9	-4	-7	-14	-5	-4	-6	-17	-8	-6	1	6	-5	-17
G	-2	-9	-3	-3	-9	-7	-4	6	-9	-11	-10	-7	-8	-9	-6	-2	-6	-15	-14	-5	-3	-5	-5	-17
H	-7	-2	0	-4	-7	1	-5	-9	9	-9	-6	-6	-10	-6	-4	-6	-7	-7	-3	-6	-1	-1	-5	-17
I	-5	-5	-5	-7	-6	-8	-5	-11	-9	8	-1	-6	-1	-2	-8	-7	-2	-14	-6	2	-6	-6	-5	-17
L	-6	-8	-7	-12	-15	-5	-9	-10	-6	-1	7	-8	1	-3	-7	-8	-7	-6	-7	-2	-9	-7	-6	-17
K	-7	0	-1	-4	-14	-3	-4	-7	-6	-6	-8	7	-2	-14	-6	-4	-3	-12	-9	-9	-2	-4	-5	-17
M	-5	-4	-9	-11	-13	-4	-7	-8	-10	-1	1	-2	11	-4	-8	-5	-4	-13	-11	-1	-10	-5	-5	-17
F	-8	-9	-9	-15	-13	-13	-14	-9	-6	-2	-3	-14	-4	9	-10	-6	-9	-4	2	-8	-10	-13	-8	-17
P	-2	-4	-6	-8	-8	-3	-5	-6	-4	-8	-7	-6	-8	-10	8	-2	-4	-14	-13	-6	-7	-4	-5	-17
S	0	-3	0	-4	-3	-5	-4	-2	-6	-7	-8	-4	-5	-6	-2	6	0	-5	-7	-6	-1	-5	-3	-17
T	-1	-6	-2	-5	-8	-5	-6	-6	-7	-2	-7	-3	-4	-9	-4	0	7	-13	-6	-3	-3	-6	-4	-17
W	-13	-2	-8	-15	-15	-13	-17	-15	-7	-14	-6	-12	-13	-4	-14	-5	-13	13	-5	-15	-10	-14	-11	-17
Y	-8	-10	-4	-11	-4	-12	-8	-14	-3	-6	-7	-9	-11	2	-13	-7	-6	-5	10	-7	-6	-9	-7	-17
V	-2	-8	-8	-8	-6	-7	-6	-5	-6	2	-2	-9	-1	-8	-6	-6	-3	-15	-7	7	-8	-6	-5	-17
B	-3	-7	6	6	-12	-3	1	-3	-1	-6	-9	-2	-10	-10	-7	-1	-3	-10	-6	-8	6	0	-5	-17
Z	-3	-4	-3	1	-14	6	6	-5	-1	-6	-7	-4	-5	-13	-4	-5	-6	-14	-9	-6	0	6	-5	-17
X	-3	-6	-3	-5	-9	-5	-5	-5	-5	-5	-6	-5	-5	-8	-5	-3	-4	-11	-7	-5	-5	-5	-5	-17
*	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	-17	1

Figure 4.6: PAM30 substitution matrix used in the CDA-based approach to find the similarity between two protein subsequences.

In this work, the substitution score between any motif instance and motif consensus was chosen to be at least s . Similar to nucleotide sequences, when constructing the subsequence network, a pair of nodes (protein subsequences of length l) were connected by an edge only if the substitution score of the two l -mers was greater than a particular threshold $s/2$ (to avoid spurious edges). After constructing the network, all possible communities of size at least q were identified in the network using the community detection algorithm, and l -mers corresponding to each community were aligned to form the motif consensus. Subsequently, each motif consensus was greedily refined towards the true motif using substitution scores as described in Section 4.2.2. The refined motif and motif instances were returned with a normalized substitution score. This process of refining community to identify motif was repeated for all communities identified by the algorithm. The top t motifs (default value of t was chosen to be 10) among all resulting motifs were then selected based on normalized substitution scores.

4.3 Hybrid Approach

4.3.1 Motivation

The BWT-based approach constructs features that occur at least twice in at least one sequence. In addition to frequency, b -mers also satisfy properties based on length and the associated suffix, as described in Section 4.1.2. Contrarily, the CDA-based approach constructs features that occur multiple times in various sequences with certain mismatches. These approaches capture different information in terms of constructing features. While the BWT-based approach does not take into account variations (possible mismatches) of the features and primarily generates non-overlapping features, the CDA-based approach generates overlapping features. A combination of the two proposed approaches, the CDA-based approach guided by BWT features, is believed to be better than the BWT and CDA-based approaches because the resulting set would include b -mers with mismatches,

thereby including features that are not limited to but similar to b -mers.

Section 4.3.2 describes the process of combining the BWT-based approach with the CDA-based approach to construct a hybrid set of features.

4.3.2 Feature Construction Using the Hybrid Approach

To combine the BWT-based approach with the CDA-based approach, b -mers (features constructed using the BWT-based approach) were provided as priors/inputs to the CDA-based approach (as shown in Figure 4.7). The only difference between the HBA approach and the CDA-based approach is the process of network construction. The process of identifying communities and motifs is identical to the CDA-based approach (Section 4.2.3).

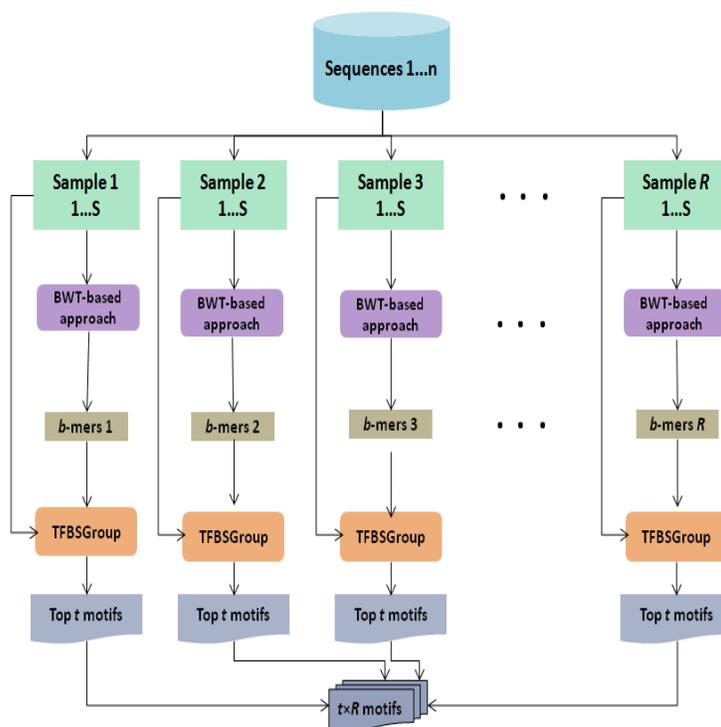


Figure 4.7: Using the hybrid approach to construct sequential features for biological sequences. Available training data containing N sequences was split into R samples, each containing S sequences. *TFBSGroup* was then invoked on each of the samples (with the c -mers guided by with the associated b -mers of each sample), and outputs of all the samples were grouped to form the final set of motifs.

Details of constructing a network of l -mers using knowledge obtained from b -mers are provided below:

Network Construction with BWT-based Features as Input: As described in Section 4.2.3, R samples were generated in which each sample consisted of S sequences. To construct the network, for each sample, instead of using all possible l -mers, l -mers were initially filtered as described below:

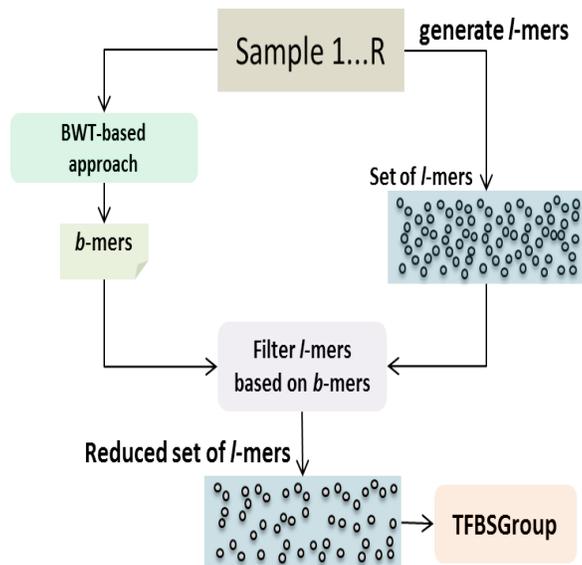


Figure 4.8: *HBA approach: The process of using b -mers as input to the CDA-based approach to identify BWT-based features with certain mismatches. The total set of l -mers in a sample were initially filtered based on similarity with any of the b -mers in that sample. Filtered l -mers were then sent as input to TFBSGroup to identify motifs.*

1. All possible b -mers within a sample were generated by providing sequences of the current sample as an input to the BWT-based approach (Section 4.1.2).
2. Each sequence in the sample was parsed to generate l -mers and retained only the l -mers that satisfied the following condition:
 - For nucleotide sequences, l -mers with a maximum Hamming distance, d_h , to at least one of the b -mers are retained. If the value of d_h is closer to d , then the

CDA-based approach is largely biased by the BWT-based approach, resulting in a similar set of features restricted to b -mers. Therefore, to avoid spurious edges and capture sufficient information from the BWT and CDA-based approaches, the value of d_h was chosen to be sufficiently larger than d (which is further used by CDA-based approach to join nodes and refine motifs).

- For protein sequences, the set of l -mers with a minimum substitution score, s_h , to at least one of the b -mers are retained. Similar to nucleotide sequences, the value of s_h was chosen to be significantly smaller than $s/2$ (where $s/2$ is used by the CDA-based approach to join nodes and s is used to refine motifs).

3. Two nodes were then joined in the resulting set, as described below:

- For nucleotide sequences, a pair of nodes (nucleotide subsequences of length l) are connected by an edge only if the Hamming distance between l -mers corresponding to the two nodes is no more than x (where $d \leq x \leq 2d$), and if they belong to different sequences.
- For protein sequences, a pair of nodes (protein subsequences of length l) are connected by an edge only if the substitution score of the two l -mers is greater than a particular threshold, $s/2$, and if they belong to different sequences.

After constructing the network, all possible communities having at least q nodes were identified, and the set of l -mers corresponding to each community was further refined to generate the final set of motifs (similar to the process of generating motifs from communities using the CDA-based approach). The resulting set of sequential features were referred to as ***h*-mers**.

Chapter 5

Experimental Setup

Section 5.1 describes the datasets used to evaluate features generated using the proposed approaches (BWT, CDA, and HBA). Section 5.2 lists high level research questions addressed throughout this work. Finally, Section 5.3, describes the experimental setup used to conduct the experiments.

5.1 Datasets

Features generated using the proposed approaches were evaluated on nucleotide and protein sequence datasets. Specific details corresponding to the datasets are provided in the following sections.

5.1.1 Alternative Splicing Datasets

For nucleotide sequences, this work focused on the problem of classifying exons as alternatively spliced or constitutive for two organisms: *C. elegans* (referred as CE) and *D. melanogaster* (referred to as DM).

- *C. elegans*:
 - **Class distribution:** The dataset consisted of 3018 sequences belonging to one of two classes: alternatively spliced (487) and constitutive (2531) exons.
 - **Generation:** The dataset was originally generated by [Rätsch et al. \[2005\]](#). The authors collected ESTs and cDNAs corresponding to *C. elegans* from Wormbase, dbEST and UniGene. ESTs were aligned to genomic DNA using BLAT, which was used to confirm exons and introns. Alignments were further refined, and all the alignments that agreed and shared at least one complete exon or intron were merged. The authors then identified pairs of sequences that shared the same 3' and 5' boundaries of the upstream and downstream exons, confirming that both sequences of each pair belonged to one gene. If the identified pair had one sequence that contained an internal exon and the other did not, then that particular exon is skipped, exhibiting alternative exon usage. The authors thus identified a total of 487 exons that exhibited alternative splicing. For constitutive exons, the authors considered exons that did not show evidence of alternative splicing, when internal exons and flanking introns were confirmed by ESTs at least twice. The authors thus extracted 2,531 exons likely to be constitutively spliced. The final dataset consisted of 487 alternatively spliced and 2531 constitutive exons along with their associated flanking introns, accounting for a total of 3018 exon triples.

- *D. melanogaster*:
 - **Class distribution:** The dataset consisted of 1409 sequences belonging to one of two classes: alternatively spliced (164) and constitutive (1245) exons.
 - **Generation:** This dataset was constructed in our lab using ALEXA [[Griffith et al., 2008](#)]. Using the precomputed dataset (various isoforms for several genes corresponding to *D. melanogaster*) obtained from ALEXA, a particular exon was

classified as alternatively spliced if it was skipped in at least one of the isoforms and constitutive otherwise.

Informative motifs that may be responsible for alternative splicing are believed to occur close to the acceptor and donor regions. Therefore, in this work, a ± 100 window around the acceptor and donor regions was used for every example in the dataset. The two regions were separated by a delimiter “&” to ignore features that span across the two regions. The process of generating each instance from the original sequence is shown in Figure 5.1. The resulting sequences of length 402 nucleotides, labeled with the class (spliced or constitutive) to which they belong were used to evaluate the features constructed using the proposed approaches.

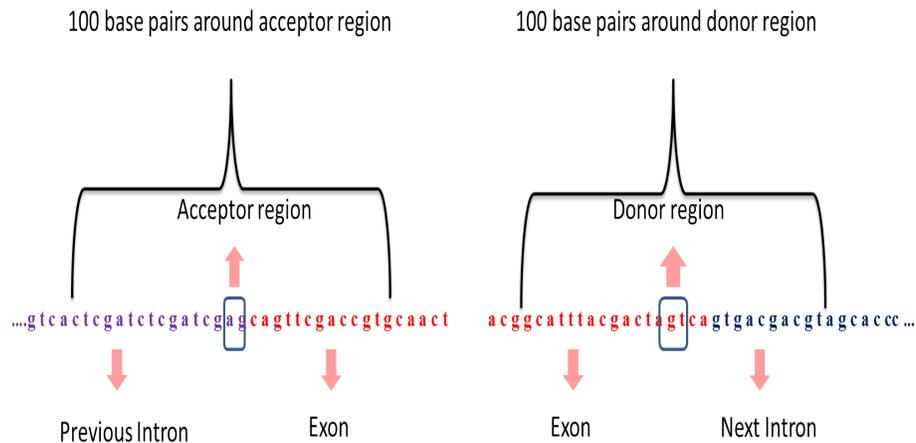


Figure 5.1: *Preprocessing of alternative splicing data. Regions of 100 bp long around the acceptor and donor regions, separated by a delimiter &, were selected.*

5.1.2 Protein Localization Datasets

For protein sequences, this work focused on classifying protein sequences based on their respective localization. Four different protein sequence datasets were used to conduct experiments: PSORTb v.2.0: Gram-negative (referred to as gNeg) and Gram-positive (referred to as gPos) datasets [Gardy et al., 2005], available online at

[http : //www.psорт.org/dataset/datasetv2.html](http://www.psорт.org/dataset/datasetv2.html), and TargetP: plant and non-plant (referred to as nonPlant) datasets [Emanuelsson et al., 2000], available online at [http : //www.cbs.dtu.dk/services/TargetP/datasets/datasets.php](http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php).

- **PSORTb datasets:**

- **Class distribution:** The gNeg dataset consisted of 1444 sequences belonging to one of five classes: cytoplasm (278), cytoplasmic membrane (309), periplasm (276), outer membrane (391), and extracellular (190). The gPos dataset consisted of 541 sequences belonging to one of the four classes: cytoplasm (194), cytoplasmic membrane (103), cellwall (61), and extracellular (183).
- **Generation:** PSORT uses knowledge obtained from signal sequences in N-terminus, transmembrane segments, cleavable signals, lipoproteins, and amino acid composition to discriminate proteins based on their localizations. PSORT uses several methods to identify signals in protein sequences [Nakai and Kanehisa, 1991; Yamaguchi et al., 1988]. PSORT then applies a set of rules based on identified signal information to label protein sequences.

- **TargetP datasets:**

- **Class distribution:** The plant dataset consisted of 940 sequences belonging to one of four classes: chloroplast (141), mitochondrial (368), secretory pathway/signal peptide (269), and other (consisting of 54 proteins labeled nuclear and 108 examples labeled cytosolic). The nonPlant dataset consisted of 2738 sequences belonging to one of three classes: mitochondrial (361), secretory pathway/signal peptide (715), and other (consisting of 1224 proteins labeled as nuclear and 438 proteins labeled as cytosolic).
- **Generation:** Protein sequences were classified based on their respective localizations using TargetP 1.1 server. TargetP predicts the subcellular location of eukaryotic protein based on the predicted presence of any N-terminal presequences:

chloroplast transit peptide (cTP), mitochondrial targeting peptide (mTP), or secretory pathway signal peptide (SP). Potential cleavage sites were also predicted for sequences predicted to contain an N-terminal presequence. TargetP uses ChloroP and SignalP to predict cleavage sites cTP and SP.

5.2 Research Questions

The primary objective of the experiments in this work was to evaluate the predictive power of features generated using the proposed approaches (b -mers, c -mers, and h -mers) compared to features generated using the sliding window-based approach (k -mers) and hashing features (referred to as r -mers) in three different scenarios: supervised, semi-supervised, and domain adaptation. Experiments were specifically motivated by the following research questions:

1. *How does the number of b -mers, c -mers, and h -mers compare to the number of all possible k -mers?* Features constructed using the proposed approaches satisfy several properties, as opposed to generating all possible subsequences of a particular length (k -mers). Therefore, dimensionality of b -mers, c -mers, and h -mers was expected to be very small compared to dimensionality of the set of all k -mers.
2. *How does the predictive power of each proposed feature set, b -mers, c -mers, and h -mers, compare to the predictive power of k -mers?* In order to investigate the predictive power of features generated using the proposed approaches (b -mers, c -mers, and h -mers), the performance of classifiers learned from sequences represented using the respective feature set was compared to the performance of classifiers learned from sequences represented using an equal number of k -mers (obtained via feature selection from the total number of k -mers). Because the proposed approaches are not supervised (i.e., do not make use of sequence labels), experiments were conducted in three learning scenarios: supervised, semi-supervised, and domain adaptation.

3. *How does the predictive power of b -mers, c -mers and h -mers compare to each other and to feature hashing?* In order to compare the predictive power of the proposed feature sets to each other as well as to another dimensionality reduction technique, specifically feature hashing, as described in Section 2.2.2, experiments were conducted in supervised, semi-supervised, and domain adaptation scenarios.

5.3 Experimental Setup: 5-fold Cross-Validation

The 5-fold cross-validation procedure was used to run all the experiments. This section details the 5-fold cross-validation scheme.

The total data was divided into five splits. Each algorithm was run five times. One of the five splits was considered for test and the remaining four splits were considered for training each time. By the end of all five runs, all the splits were used as test data one time. Performance was reported by taking the average of the predictions obtained in each of the five runs. Figure 5.2 shows the splits of the total data and the corresponding train and test data. The cross-validation procedure was used to reduce the bias in data sampling.

Supervised learning scenario: In the supervised learning scenario, we used all train data (four folds) as labeled in order to train the classifier. The trained classifier was then used to predict the test data (fifth fold). Figure 5.3 shows the split of train and test data for one of the five iterations.

Semi-supervised learning scenario: SSL is mainly used when the amount of available labeled data is small, but large amounts of unlabeled data are available. To ensure the data assumption of SSL, the data was split as follows:

- One of the five folds, 20% of the total data, was used to test the classifier (T).
- Data from the remaining four folds was split into:
 - 10% as labeled data used for training (L)

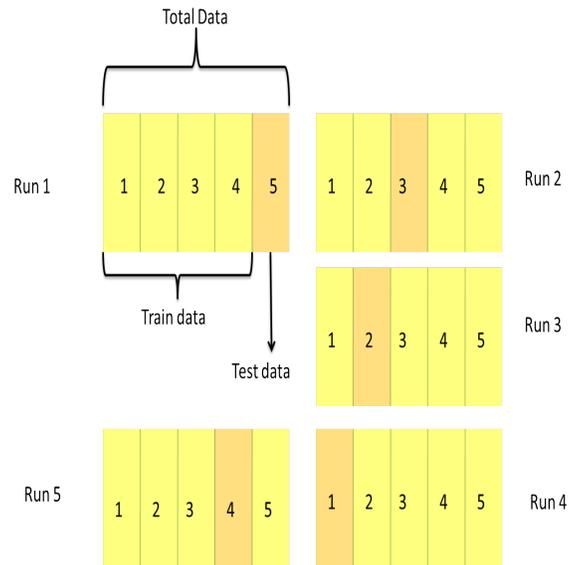


Figure 5.2: *The 5-fold cross-validation setting for experiments. On each run, four folds were used for training and one fold was used for testing.*

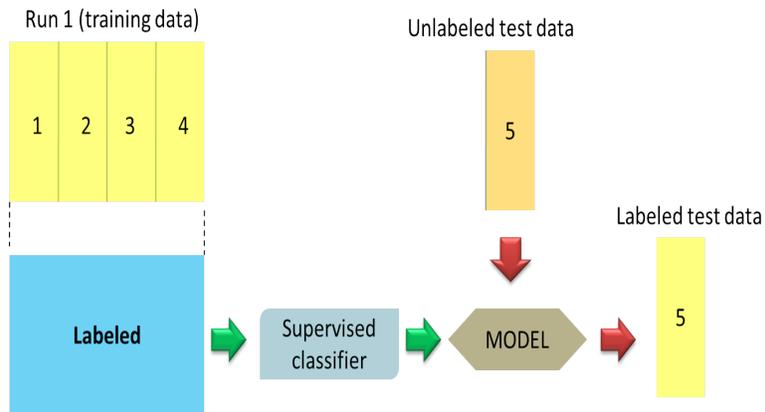


Figure 5.3: *One of the five runs of the 5-fold cross-validation setting in a supervised learning scenario. Total data was divided into four folds for training and one fold for testing. All train data was used as labeled to learn a supervised classifier, which was later used to predict the test data.*

– 90% as unlabeled data for training (U):

- * In order to study variation of the performance with unlabeled data, the amount of unlabeled data was varied from 20% to 90% with increments of 10% (while maintaining the amount of labeled data fixed to 10%)

The amount of labeled data was chosen to be smaller than the amount of unlabeled data. On each of the five iterations corresponding to the cross-validation setting, an SSL classifier (ST or CT, as described in Section 2.2.1) was learned using labeled and unlabeled data ($L + U$) and further used to predict respective test data (T), as shown in Figure 5.4.

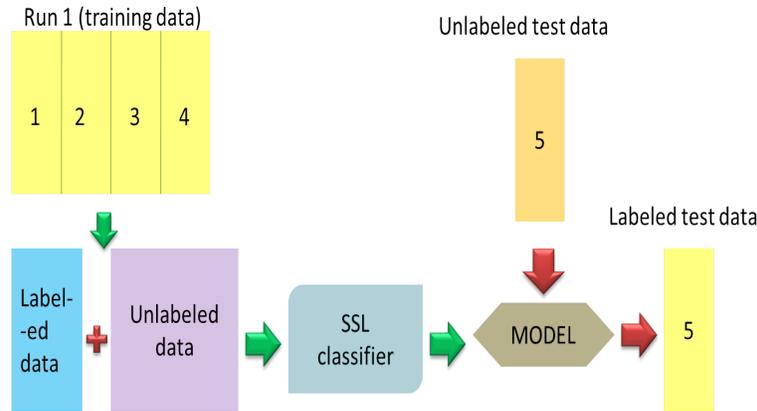


Figure 5.4: One of the five runs of the 5-fold cross-validation setting in an SSL scenario. Total data was split into four folds for training and one fold for testing, and the train data was further split into labeled and unlabeled data and sent as input to the SSL classifier.

Domain adaptation scenario: As discussed in Section 2.2.1, in order to evaluate features in the domain adaptation framework, data was required from two different domains, source and target. As shown in Figure 5.5, on each iteration, all the data from source domain was used for training (referred to as sL), while data from the target domain was split into three subsets:

- One of the five folds, 20% of the total target data, was used to test the classifier (referred to as tT).
- Data from the remaining four folds of target data was further split into:

- 20% as labeled for training (referred to as tL):
- 80% as unlabeled for training (referred to as tU)
- * The amount of unlabeled data was varied from 20% to 80% with increments of 20%, i.e., 20%, 40%, 60% and 80%

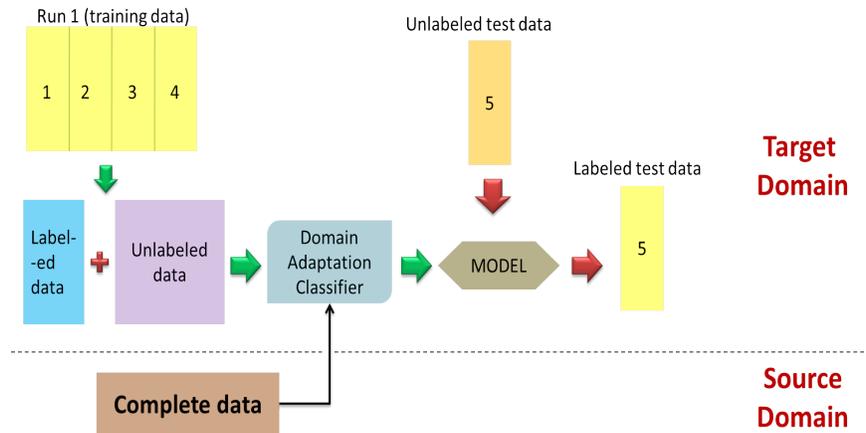


Figure 5.5: One of the five folds of the 5-fold cross-validation setting in a domain adaptation scenario, containing data from two domains (source and target). Complete data from the source domain was treated as labeled, while target data was further split into labeled and unlabeled (four folds) and test data (one fold).

5.4 Learning Algorithms and Other Experimental Details

As discussed, experiments were conducted in three learning scenarios. The following default learning algorithms were used for various scenarios:

- Supervised learning scenario: NBM (Section 2.2.1).
- Semi-supervised learning scenario: ST and CT, with NBM as the base classifier (Section 2.2.1). CT is a two-view learning process in which each view corresponds to a

subset of features. To generate views, the features were randomly split into two subsets. The number of iterations of both ST and CT were fixed to 100, and sample size was fixed to 50.

- Domain adaptation scenario: Multinomial Naïve Bayes for domain adaptation (Section 2.2.1).

Experiments were conducted on all six datasets (CE, DM, gPos, gNeg, plant, and non-Plant) in supervised, SSL and domain adaptation scenarios. For the domain adaptation scenario, a source domain with labeled data was assumed to be available in addition to the target domain labeled and unlabeled data. Experiments were conducted with the following pairs of source→target domains: $CE \rightarrow DM$, $DM \rightarrow CE$, $GP \rightarrow GN$, $GN \rightarrow GP$, $P \rightarrow NP$, and $NP \rightarrow P$, respectively. Only overlapping classes within a pair of domains were used in the domain adaptation scenario (i.e., cytoplasm, cytoplasmic membrane, and extracellular for the GP/GN pairs, and mitochondrial, secretory pathway/signal peptide, and others for P/NP pairs). For each run, all data from the source domain was used as labeled (sL), and the four target train data folds were split into labeled (tL) and unlabeled (tU). In all the experiments conducted, area under ROC curve (AUC) was used as a performance metric.

Chapter 6

Burrows Wheeler Transform

Approach: Experiments and Results

Section 6.1 of this chapter lists the set of more specific research questions related to the BWT-based approach proposed in this work. Section 6.2 details the default parameters used and experiments performed to address research questions. Finally, Section 6.3 includes discussion and analysis of results.

6.1 Research Questions

1. *How does the number of b -mers compare to the number of k -mers for various biological sequence data sets?*

The BWT-based approach only returns a subset of k -mers that occur at least twice in at least one sequence. However, not all k -mers that occur at least twice are returned by the BWT-based approach because, in addition to frequency, b -mers satisfy additional properties based on suffix and length (Section 4.1). Therefore, dimensionality of the set of b -mers was expected to be very small compared to dimensionality of the set of k -mers, thereby making the BWT-based approach an attractive dimensionality

reduction technique.

2. *When feature selection is used in the case of supervised learning, which set, the set of b -mers or the set of k -mers, is more informative?*

In the supervised scenario, availability of labeled data can be sufficient to accurately estimate feature-class dependencies. Therefore, good performance may be attained with a small number of highly informative features obtained from the feature selection technique. In order to evaluate features in the b -mers and k -mers sets, feature selection was applied to select the same number of features from both feature sets. Classifiers were then learned from each subset, respectively, and their performances were compared.

Ideally, good performance should be achieved with a sufficiently small number of informative features. Therefore, the number of selected features from b -mers and k -mers was varied and the respective performance of the classifiers was recorded.

3. *How does the predictive power of b -mers compare to the predictive power of k -mers in a semi-supervised learning scenario?*

In SSL, feature selection techniques may not accurately capture feature-class dependencies, thereby selecting uninformative features. Because the dimensionality of b -mers is much smaller than dimensionality of k -mers, and considering the unsupervised nature of the proposed approach in generating features, the predictive power of b -mers was investigated in an SSL scenario.

4. *How does the predictive power of b -mers compare to the predictive power of k -mers in the domain adaptation scenario?*

For domain adaptation, as discussed in Section 5.3, data was obtained from two different domains: source domain and target domain. The predictive power of b -mers was investigated in a domain adaptation learning scenario when the amount of available labeled data from target domain is small.

5. *How do results vary with amounts of unlabeled data in semi-supervised and domain adaptation scenarios?*

The total data (labeled, unlabeled and test for SSL scenario; target labeled, target unlabeled, target test and source labeled for domain adaptation scenario) was represented using b -mers and k -mers. Intuitively, for the same amount of labeled data, increasing the amount of unlabeled data is expected to benefit the classifier. To study this behavior, the amount of labeled data (target labeled in domain adaptation scenario) was fixed in the experiments and the amount of unlabeled data (target unlabeled in domain adaptation scenario) was varied to observe variation in the performance of respective classifiers (SSL and domain adaptation).

6.2 Parameters and Experiments

6.2.1 Default Parameters

The following default values were used for parameters of the BWT-based approach when conducting experiments:

- Length of features: Feature length in the BWT-based approach was controlled by parameter l . From the total set of variable length features obtained from the BWT-based approach, features with length l were filtered, where l can be of variable length. In this work, the following values of l were used for nucleotide and protein sequences.
 - Nucleotide sequences: 6, 7, and 8.
 - Protein sequences: 2, 3, and 4.
- Minimum number of occurrences, r : This parameter controlled the minimum number of times resulting b -mers occurred in at least one sequence. In this work, r was chosen to be 2, indicating that resulting features occurred at least twice in at least one sequence.

6.2.2 Experiments

This section, describes the set of experiments conducted to address research questions corresponding to BWT-based features discussed in Section 6.1.

Supervised scenario: As discussed in Section 5.3, total data was divided into train and test data (5-fold cross-validation setting). In each iteration (run), train data was used to generate the set of features (b -mers and k -mers) as described below:

- **Generating b -mers:** To generate b -mers, the BWT-based approach (described in Section 4.1.2) was invoked with train sequences, length of features (l) and minimum number of occurrences of the motif in the original sequence (r) as input. The BWT-based approach returned the set of b -mers. The number of b -mers was denoted by $D_{b\text{-mers}}$.
- **Generating k -mers:** The sliding window-based approach (Section 4) was used, with $k = l$ in order to perform a fair comparison with the proposed approach. The number of k -mers was denoted by $D_{k\text{-mers}}$.

In order to generate features of variable length, the above process was repeated for multiple values of l . The first question from Section 6.1 was addressed by comparing $D_{b\text{-mers}}$ to $D_{k\text{-mers}}$.

In order to address the second question from Section 6.1, feature selection was applied on the labeled data (all the train data in supervised scenario), represented using b -mers (referred to as $Lab_{b\text{-mers}}$) and k -mers (referred to as $Lab_{k\text{-mers}}$), separately, to select top f features. Let $f_{b\text{-mers}}$ be top f features selected from $Lab_{b\text{-mers}}$, and $f_{k\text{-mers}}$ be top f features selected from $Lab_{k\text{-mers}}$. The performance of the classifiers learned from the data represented using $f_{b\text{-mers}}$ and $f_{k\text{-mers}}$ was compared. Although dimensionality was reduced to f for both b -mers and k -mers, obviously, the features in $f_{b\text{-mers}}$ differed from the features in $f_{k\text{-mers}}$. f was varied from 50 to 1500 (specifically, 50, 100, 150, 200, 250, 500, 1000, or 1500).

Semi-supervised scenario: The unsupervised nature of the BWT-based approach to generate features motivated use of b -mers in an SSL scenario. In the semi-supervised scenario, as discussed in Section 5.3 the total data was split into labeled, unlabeled, and test, such that the amount of unlabeled data was larger than the amount of labeled data.

As class labels are not taken into account when generating b -mers and k -mers, all the train data (labeled and unlabeled) was used to generate b -mers (as described in Section 4.1.2) and k -mers. In order to perform a fair comparison, feature selection was used on labeled data only, represented using k -mers in order to select top $D_{b\text{-mers}}$ features. Labeled, unlabeled, and test data were represented using b -mers and k -mers (reduced to the dimensionality of $D_{b\text{-mers}}$). The transformed labeled and unlabeled data were used to learn semi-supervised classifiers which were further used to predict corresponding test data for both feature sets; performance of the classifiers was recorded. In order to understand the predictive power of features in a semi-supervised scenario, the amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%, thereby addressing the third and fifth questions from Section 6.1.

Domain adaptation scenario: Because class labels were not taken into account, all the train data (labeled and unlabeled) from target domain was used to generate b -mers (as described in Section 4.1.2) and k -mers. Similar to the experimental setup for the SSL scenario, feature selection was used on target labeled data (tL) represented using k -mers to select top $D_{b\text{-mers}}$ features. Source labeled (sL), target labeled (tL), target unlabeled (tU), and target test (tT) data were then represented using both b -mers and k -mers (reduced to the dimensionality of $D_{b\text{-mers}}$). Domain adaptation classifier (described in Section 2.2.1) was trained using source labeled, target labeled, and target unlabeled data, represented using b -mers and the reduced set of k -mers. The classifier was then used to predict target test data (T), and AUC was recorded. The amount of target labeled data was varied from 5% to 20% and target unlabeled data was varied from 20% to 80%, thereby addressing the fourth and fifth questions from Section 6.1.

Table 6.1: Comparison of the number of features generated using b -mers and k -mers for the six datasets used, averaged over 5 folds.

Dataset	b -mers	k -mers
CE	9276	84158
DM	6669	84513
gPos	4696	86319
gNeg	6354	117935
plant	4498	103611
nonPlant	13008	151261

6.3 Results

6.3.1 Dimensionality Comparison

Table 6.1 presents the number of features generated using the sliding window-based approach (k -mers) and BWT-based approach (b -mers), averaged over five folds. This experiment was performed in a supervised learning scenario in which all available data (labeled data) was used for generating features. As shown in Table 6.1, the number of features in the set of k -mers was significantly greater than the number of features in the set of b -mers.

6.3.2 Supervised Scenario: b -mers versus k -mers

This section, evaluates the predictive power of b -mers in a supervised learning scenario. As discussed in Section 6.2.2, feature selection using all available labeled data was used to select top f features (from b -mers and k -mers, respectively) that were further used to represent the sequences. NBM classifiers (Section 2.2.1) were then learned using all available labeled (train) data. The resulting classifiers were used to predict test data, and performances on each fold were recorded and averaged. This process was repeated for different values of f (varied from 50 to 1500), and corresponding AUC values (averaged over five folds) were reported.

Table 6.2 reports AUC values of the NBM classifier learned using the two sets of features, b -mers and k -mers, for the two nucleotide (CE and DM), and four protein (gPos, gNeg, plant,

and nonPlant) datasets. For each dataset, maximum AUC between b -mers and k -mers for each variation of the number of features selected was reported in the **bold** font. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k -mers, AUC values of the NBM classifier learned using all the k -mers are also reported in Table 6.2, referred to as k -mersAll (*italicized*). To better understand the behavior, results are also plotted in Figure 6.1. As shown in Table 6.2, in 46 out of 48 experiments, the NBM classifier learned from b -mers performed better than the classifier learned from k -mers. The remaining 2 experiments corresponded to nonPlant dataset, for which class distribution was 14:26:**60** (mitochondrial:secretory pathway/signal peptide:nuclear+cytosolic). The data was skewed to the third class (nuclear+cytosolic) as opposed to other datasets. Because the BWT-based approach constructs features from each sequence individually, total features constructed from the train sequences (skewed to one class) are also believed to be biased to capture information corresponding to the skewed class. As a result, a majority of b -mers probably capture information corresponding to the third class for the nonPlant dataset.

Furthermore, as shown in Figure 6.1, for nucleotide sequences, the classifiers learned from b -mers outperformed the classifiers learned from all k -mers. Although b -mers are a subset of k -mers, and yet the predictive power of b -mers is better than that of k -mers, suggesting that the BWT-based approach successfully filtered uninformative k -mers that are acting as outliers. For protein sequences, it can be observed that the performance of the classifier learned using b -mers is comparable to the performance of the classifier learned using all k -mers for large values of f . This suggests that a considerable (or even better, in case of nucleotide sequences) performance is achieved with a relatively small set of features.

For small f , the set of k -mers includes features that are most informative for the class according to feature selection criterion; but probably many of these features are variants of each other (i.e., features with overlaps). Consequently, although most informative features were included, not all informative features were included (meaning that variations of an informative feature covered much of the set and other informative features were not selected).

Table 6.2: Variation of the performance of NBM classifier with the number of b -mers and k -mers features selected in supervised learning scenario. Table includes AUC values for CE, DM, gPos, gNeg, plant and nonPlant datasets. For each dataset, maximum AUC between b -mers and k -mers for each variation of the number of features selected is reported in **bold** font. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k -mers, AUC values of the NBM classifier learned using all the k -mers (denoted by k -mersAll) are also presented with italic font.

Index	1	2	3	4	5	6	7	8
Number of features	50	100	150	200	250	500	1000	1500
CE(b -mers)	0.579	0.676	0.722	0.735	0.741	0.797	0.821	0.825
CE(k -mers)	0.554	0.566	0.576	0.578	0.584	0.668	0.684	0.684
CE(k -mersAll)	<i>0.76</i>							
DM(b -mers)	0.536	0.667	0.704	0.699	0.701	0.71	0.72	0.719
DM(k -mers)	0.41	0.398	0.398	0.387	0.386	0.393	0.46	0.526
DM(k -mersAll)	<i>0.566</i>							
gPos(b -mers)	0.664	0.714	0.75	0.784	0.801	0.849	0.884	0.902
gPos(k -mers)	0.649	0.698	0.736	0.77	0.786	0.838	0.86	0.87
gPos(k -mersAll)	<i>0.951</i>							
gNeg(b -mers)	0.691	0.753	0.797	0.829	0.861	0.91	0.932	0.939
gNeg(k -mers)	0.623	0.667	0.697	0.711	0.742	0.81	0.883	0.893
gNeg(k -mersAll)	<i>0.942</i>							
plant(b -mers)	0.65	0.73	0.757	0.765	0.775	0.824	0.846	0.86
plant(k -mers)	0.533	0.569	0.577	0.595	0.599	0.652	0.721	0.755
plant(k -mersAll)	<i>0.829</i>							
nonPlant(b -mers)	0.663	0.701	0.744	0.793	0.802	0.838	0.853	0.861
nonPlant(k -mers)	0.655	0.7	0.734	0.759	0.783	0.827	0.856	0.867
nonPlant(k -mersAll)	<i>0.841</i>							

Contrarily, the set of b -mers consisted of more informative features compared to k -mers, as some variants may be excluded, and thus potentially cause the set of b -mers to result in better performance as compared to k -mers for relatively small number of features (50 to 1500).

Based on all observations, the BWT-based approach successfully reduced the initial number of features by retaining most of the informative features, thereby representing an effective dimensionality reduction technique, even for supervised learning when sufficient labeled data is available.

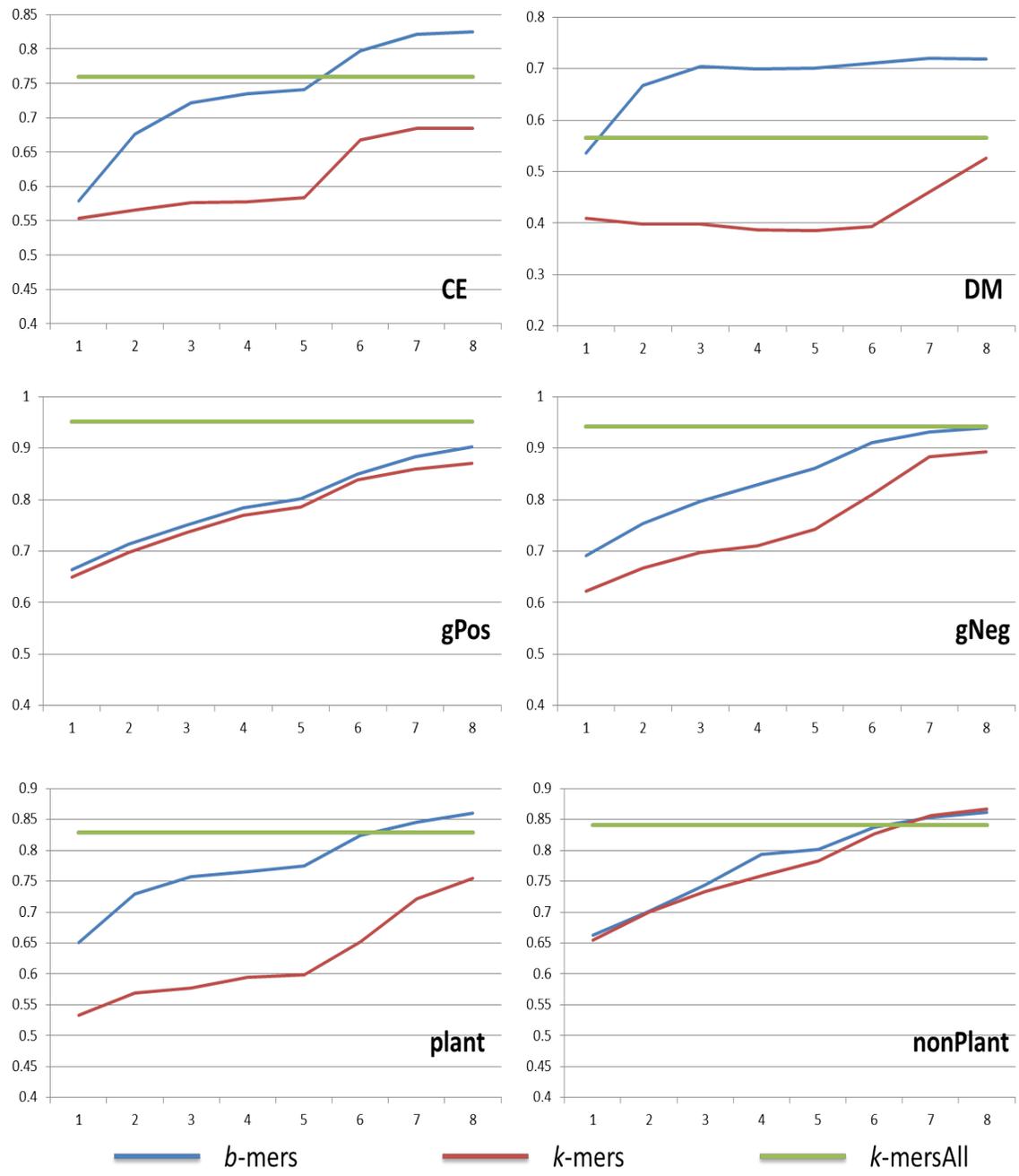


Figure 6.1: Variation of the performance of NBM classifier with the number of b-mers and k-mers selected in supervised learning scenario is shown in the following graphs. Besides these, the performance of NBM classifier when all the k-mers are used as features is also shown in the graphs (k-mersAll). In each graph, x-axis represents the number of features selected from each feature set, f , which is varied from 50 to 1500 specifically. y-axis represents the AUC values of NBM classifier learned from the respective feature sets. See Table 6.2 for the numbers shown on x-axis and the corresponding number of features selected, f .

6.3.3 Semi-supervised Scenario: *b*-mers versus *k*-mers

As discussed in Section 5.3, available train data was split into various combinations of labeled and unlabeled data. For each combination, ST and CT classifiers were learned for nucleotide (CE and DM) and protein (gPos, gNeg, plant, and nonPlant) sequences represented using *b*-mers and *k*-mers (reduced to the number of *b*-mers using feature selection). Besides using the reduced number of *k*-mers, to get an estimate of the best AUC that can be achieved using *k*-mers, AUC values of ST and CT classifiers learned using all the *k*-mers are also reported in Table 6.2, referred to as *k*-mersAll (*italicized*).

In order to specifically address the fifth question from Section 6.1, the amount of labeled data, L , was fixed to 10% and the amount of unlabeled data, U , was varied as follows: {20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%}.

Table 6.3 reports AUC values of the set of experiments conducted in SSL scenario. For each dataset, maximum AUC between *b*-mers and *k*-mers for each variation of the amount of unlabeled data is reported in **bold** font. To better understand the behavior, results are also plotted in Figures 6.2 and 6.3. Results in the table indicate that *b*-mers outperformed *k*-mers in 41 out of 48 cases with ST and 48 out of 48 cases with CT, suggesting that with small amounts of labeled data, feature selection was unable to accurately capture the feature-class dependency scores, thus selecting a set of uninformative features (*k*-mers).

Furthermore, as shown in Figures 6.2 and 6.3, in majority of the cases, the classifier learned from *b*-mers outperformed the classifier learned from all *k*-mers. When the amount of available labeled data is small, using large set of features can mislead the classifier, as there is no sufficient information to capture the knowledge from all the features (especially, when some of them act as outliers).

When the amount of unlabeled data increased, intuitively, the performance of any SSL classifier could be expected to improve because the more the unlabeled data, the better is the classifier trained at each iteration. However, additional unlabeled data can occasionally result in mislabeled data being used for training, thereby degrading the classifier performance.

Therefore, performance may initially increase but decrease with the increased amounts of unlabeled data, as observed in most cases when CT was used as the semi-supervised classifier. However, in most of the cases with ST, the performance decreased even with a small increase in the amount of unlabeled data, with the exception of DM. Furthermore, when CT was used, a maximum AUC for four out of the six datasets (DM, gPos, gNeg, and plant datasets) was obtained, and values for the other two datasets were also comparable to ST, suggesting that CT more efficiently used available unlabeled data as compared to ST.

6.3.4 Domain Adaptation Scenario: *b*-mers versus *k*-mers

Table 6.4 and Figure 6.4 shows AUC values over the 5-fold cross-validation for domain adaptation algorithm described in Section 2.2.1 in six source and target configurations using varying amounts of target unlabeled data (table columns) and *b*-mers or *k*-mers as features (table rows). For each comparison, the largest AUC value (between *b*-mers and *k*-mers) is highlighted with bold font. For example, the row *CE* \rightarrow *DM* with 20% target unlabeled data reports AUC values when using *b*-mers, 0.7088, and *k*-mers, 0.5778, with 0.7088 in bold font because it is greater than 0.5778 when training the algorithm on the CE source domain and DM target domain. Besides using the reduced number of *k*-mers, to get an estimate of the best AUC that can be achieved using *k*-mers, AUC values of domain adaptation classifier learned using all the *k*-mers are also reported in Table 6.4, referred to as *k*-mersAll (*italicized*).

The classifier had higher AUC in 22 out of 24 cases when using *b*-mer features compared to when using *k*-mer features.

However, the more target unlabeled data, the lower the classifier’s accuracy. This behavior seems using unlabeled data helps the classifier, but too much unlabeled data misleads the classifier. For the distance between source and target domains, the intuition is that the more closely related domains, are the higher the classifier’s accuracy, as evidenced by the results from this study that showed higher AUC values for bacteria organisms, which are

Table 6.3: Variation of the performance with the amount of unlabeled data used with *ST* and *CT* classifiers, learned using *b*-mers and *k*-mers (reduced to the number of *b*-mers), respectively. Table includes AUC values for *CE*, *DM*, *gPos*, *gNeg*, *plant*, and *nonPlant* datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90%. For each dataset, maximum AUC between *b*-mers and *k*-mers for each variation of the amount of unlabeled data is reported in **bold** font. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using *k*-mers, AUC values of the *ST* and *CT* classifiers learned using all the *k*-mers (denoted by *k*-mersAll) are also presented with *italic* font.

Self-training								
Index	1	2	3	4	5	6	7	8
Unlabeled	20%	30%	40%	50%	60%	70%	80%	90%
<i>CE</i> (<i>b</i> -mers)	0.673	0.663	0.661	0.659	0.665	0.664	0.663	0.658
<i>CE</i> (<i>k</i> -mers)	0.508	0.576	0.603	0.622	0.609	0.606	0.601	0.587
<i>CE</i> (<i>k</i> -mersAll)	<i>0.548</i>	<i>0.547</i>	<i>0.545</i>	<i>0.547</i>	<i>0.548</i>	<i>0.545</i>	<i>0.545</i>	<i>0.548</i>
<i>DM</i> (<i>b</i> -mers)	0.621	0.606	0.596	0.623	0.636	0.621	0.635	0.63
<i>DM</i> (<i>k</i> -mers)	0.457	0.452	0.447	0.444	0.457	0.424	0.531	0.524
<i>DM</i> (<i>k</i> -mersAll)	<i>0.517</i>	<i>0.516</i>						
<i>gPos</i> (<i>b</i> -mers)	0.864	0.858	0.825	0.825	0.808	0.792	0.782	0.768
<i>gPos</i> (<i>k</i> -mers)	0.627	0.622	0.62	0.632	0.622	0.626	0.619	0.627
<i>gPos</i> (<i>k</i> -mersAll)	<i>0.723</i>	<i>0.684</i>	<i>0.703</i>	<i>0.673</i>	<i>0.673</i>	<i>0.674</i>	<i>0.685</i>	<i>0.666</i>
<i>gNeg</i> (<i>b</i> -mers)	0.875	0.857	0.825	0.828	0.833	0.798	0.791	0.826
<i>gNeg</i> (<i>k</i> -mers)	0.84	0.841	0.842	0.853	0.843	0.848	0.839	0.762
<i>gNeg</i> (<i>k</i> -mersAll)	<i>0.734</i>	<i>0.728</i>	<i>0.724</i>	<i>0.722</i>	<i>0.721</i>	<i>0.719</i>	<i>0.719</i>	<i>0.716</i>
<i>plant</i> (<i>b</i> -mers)	0.727	0.712	0.695	0.712	0.712	0.715	0.709	0.71
<i>plant</i> (<i>k</i> -mers)	0.607	0.61	0.63	0.639	0.643	0.645	0.642	0.641
<i>plant</i> (<i>k</i> -mersAll)	<i>0.613</i>	<i>0.606</i>	<i>0.602</i>	<i>0.6</i>	<i>0.599</i>	<i>0.597</i>	<i>0.596</i>	<i>0.595</i>
<i>nonPlant</i> (<i>b</i> -mers)	0.803	0.778	0.752	0.724	0.717	0.684	0.639	0.65
<i>nonPlant</i> (<i>k</i> -mers)	0.691	0.686	0.681	0.671	0.668	0.667	0.666	0.663
<i>nonPlant</i> (<i>k</i> -mersAll)	<i>0.582</i>	<i>0.579</i>	<i>0.576</i>	<i>0.575</i>	<i>0.573</i>	<i>0.573</i>	<i>0.572</i>	<i>0.571</i>
Co-training								
Index	1	2	3	4	5	6	7	8
Unlabeled	20%	30%	40%	50%	60%	70%	80%	90%
<i>CE</i> (<i>b</i> -mers)	0.671	0.656	0.658	0.658	0.65	0.644	0.666	0.668
<i>CE</i> (<i>k</i> -mers)	0.497	0.573	0.614	0.654	0.621	0.62	0.628	0.64
<i>CE</i> (<i>k</i> -mersAll)	<i>0.549</i>	<i>0.545</i>	<i>0.545</i>	<i>0.545</i>	<i>0.546</i>	<i>0.546</i>	<i>0.545</i>	<i>0.544</i>
<i>DM</i> (<i>b</i> -mers)	0.607	0.608	0.605	0.609	0.631	0.622	0.637	0.624
<i>DM</i> (<i>k</i> -mers)	0.429	0.436	0.455	0.45	0.44	0.413	0.536	0.526
<i>DM</i> (<i>k</i> -mersAll)	<i>0.517</i>	<i>0.516</i>						
<i>gPos</i> (<i>b</i> -mers)	0.849	0.859	0.876	0.866	0.85	0.863	0.866	0.838
<i>gPos</i> (<i>k</i> -mers)	0.623	0.62	0.619	0.616	0.625	0.622	0.627	0.62
<i>gPos</i> (<i>k</i> -mersAll)	<i>0.705</i>	<i>0.707</i>	<i>0.697</i>	<i>0.666</i>	<i>0.687</i>	<i>0.68</i>	<i>0.668</i>	<i>0.681</i>
<i>gNeg</i> (<i>b</i> -mers)	0.886	0.903	0.873	0.868	0.859	0.856	0.855	0.877
<i>gNeg</i> (<i>k</i> -mers)	0.846	0.844	0.844	0.846	0.849	0.838	0.819	0.752
<i>gNeg</i> (<i>k</i> -mersAll)	<i>0.734</i>	<i>0.728</i>	<i>0.724</i>	<i>0.722</i>	<i>0.722</i>	<i>0.719</i>	<i>0.717</i>	<i>0.717</i>
<i>plant</i> (<i>b</i> -mers)	0.746	0.72	0.719	0.705	0.715	0.713	0.716	0.687
<i>plant</i> (<i>k</i> -mers)	0.604	0.606	0.634	0.637	0.647	0.642	0.643	0.647
<i>plant</i> (<i>k</i> -mersAll)	<i>0.613</i>	<i>0.606</i>	<i>0.602</i>	<i>0.601</i>	<i>0.599</i>	<i>0.598</i>	<i>0.596</i>	<i>0.595</i>
<i>nonPlant</i> (<i>b</i> -mers)	0.801	0.792	0.785	0.78	0.772	0.755	0.744	0.726
<i>nonPlant</i> (<i>k</i> -mers)	0.691	0.687	0.681	0.672	0.666	0.664	0.664	0.662
<i>nonPlant</i> (<i>k</i> -mersAll)	<i>0.582</i>	<i>0.579</i>	<i>0.576</i>	<i>0.574</i>	<i>0.573</i>	<i>0.573</i>	<i>0.572</i>	<i>0.571</i>

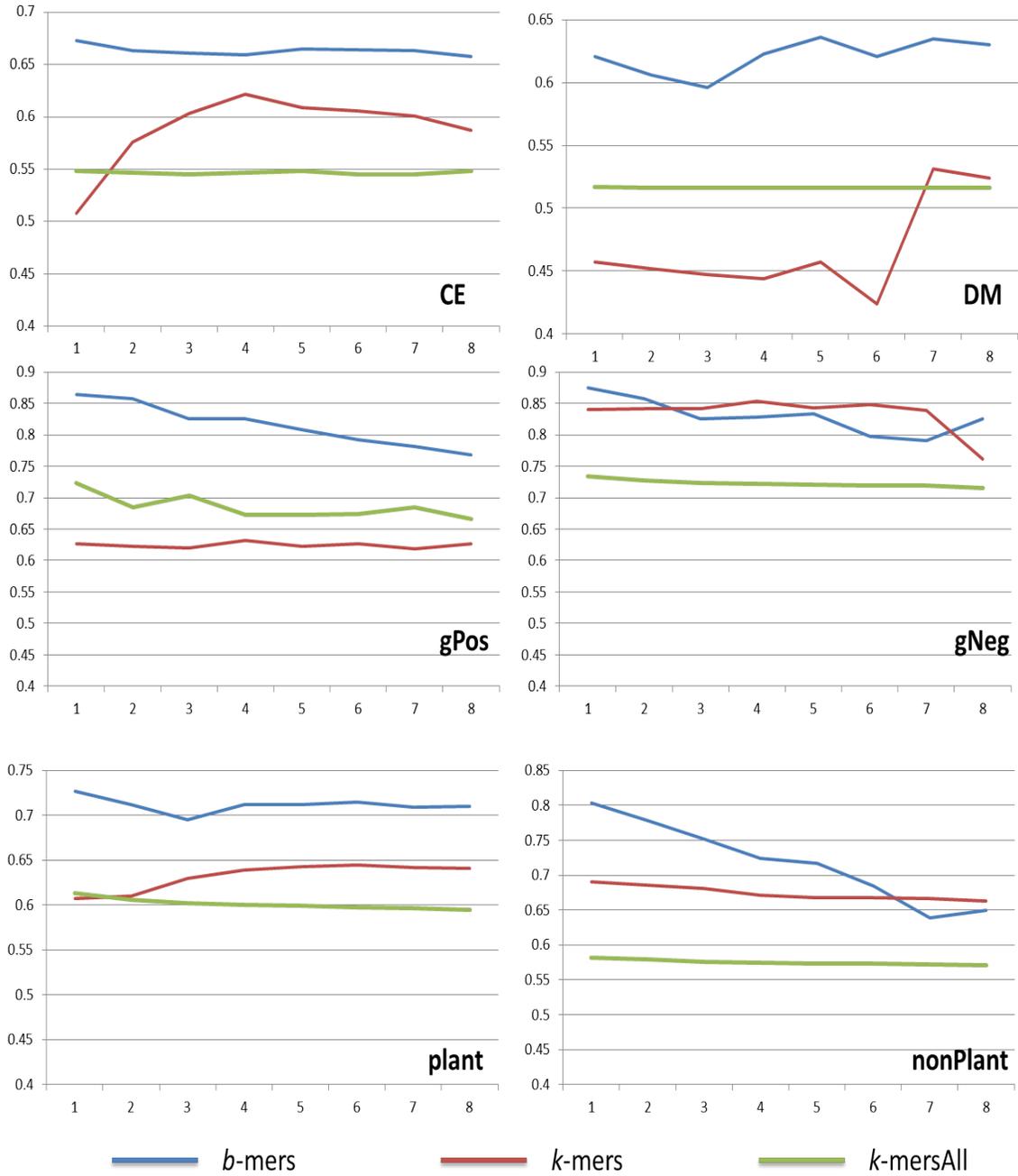


Figure 6.2: Variation of the performance with the amount of unlabeled data used with ST classifier, learned using b-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on x-axis). See Table 6.3 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

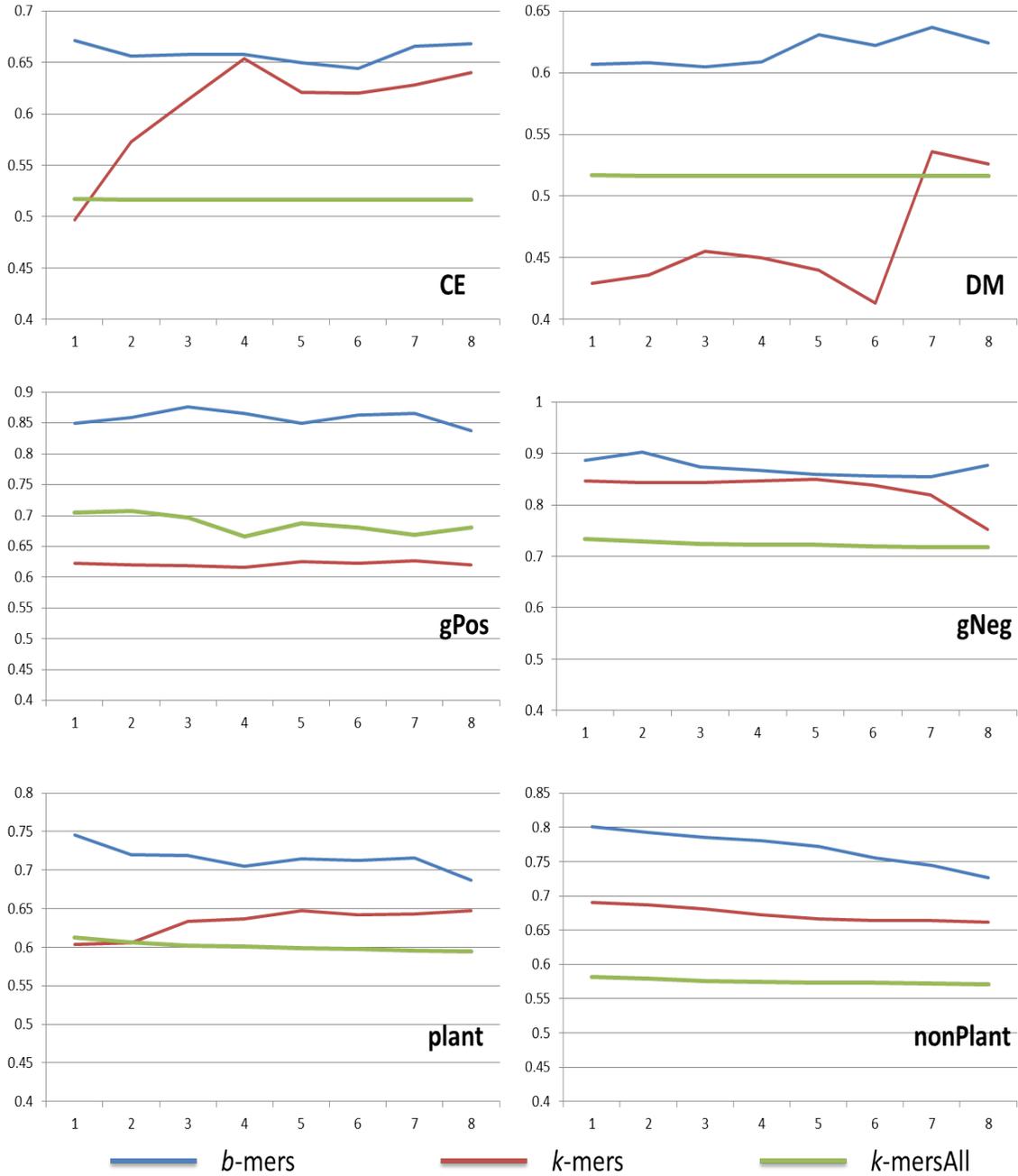


Figure 6.3: Variation of the performance with the amount of unlabeled data used with CT classifier, learned using b-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on x-axis). See Table 6.3 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

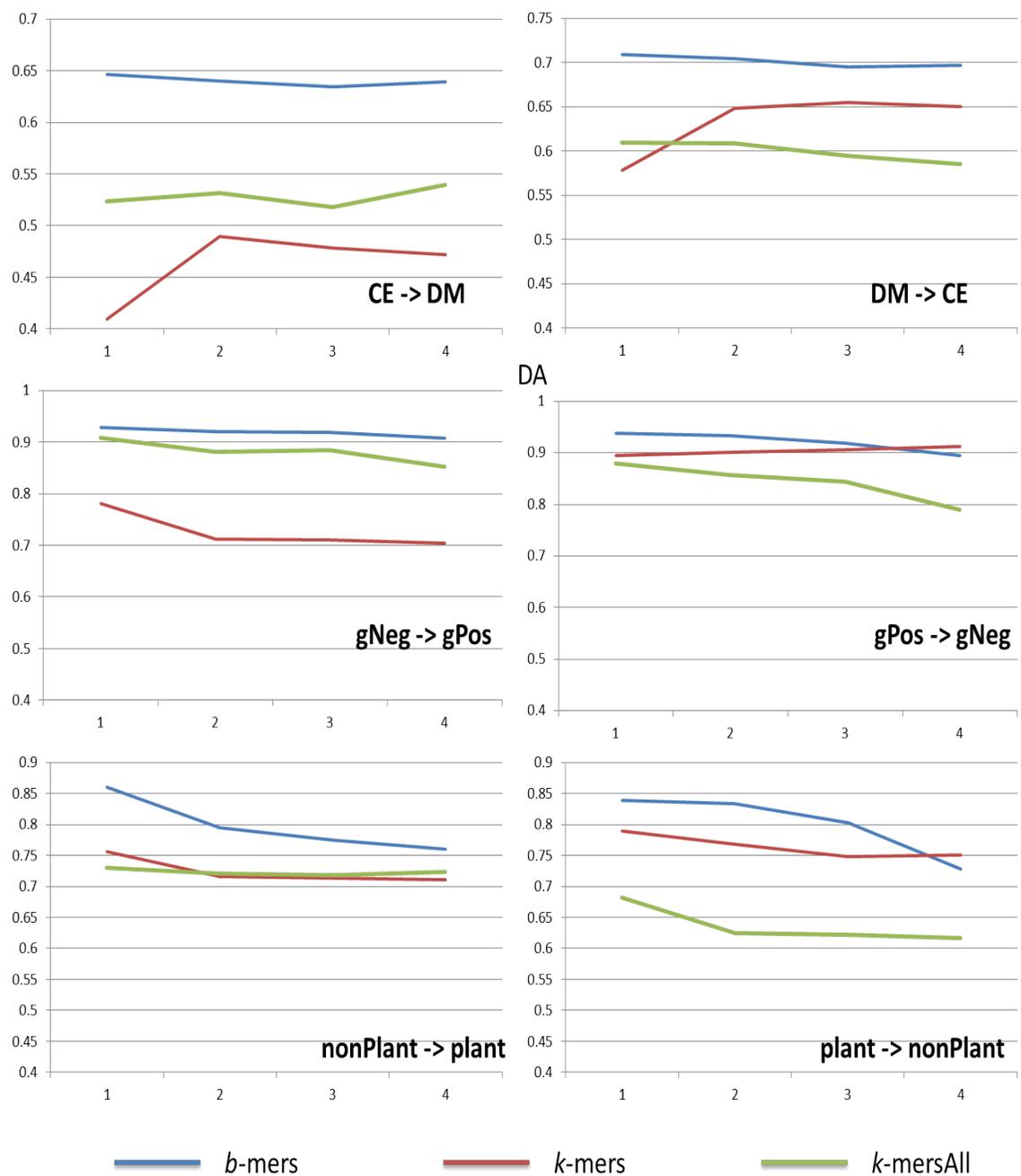


Figure 6.4: Variation of the performance with the amount of unlabeled data used with NBM for domain adaptation classifier, learned using b-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE→DM, DM→CE, gPos→gNeg, gNeg→gPos, plant→nonPlant, and nonPlant→plant combinations of source→target datasets when the amount of labeled data was fixed to 20%, while the amount of unlabeled data was varied from 20% to 80% (on x-axis). See Table 6.4 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

Table 6.4: Variation of the performance with the amount of unlabeled data used with NBM for domain adaptation classifier, learned using b-mers and k-mers (reduced to the number of b-mers), respectively. Table includes AUC values for the following combinations of source and target datasets: $CE \rightarrow DM$, $DM \rightarrow CE$, $gPos \rightarrow gNeg$, $gNeg \rightarrow gPos$, $plant \rightarrow nonPlant$, and $nonPlant \rightarrow plant$. For each combination of the dataset, the amount of labeled and unlabeled data used and the largest AUC value (between b-mers and k-mers) is highlighted with **bold** font. For all experiments, the amount of labeled data is fixed to 20%. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k-mers, AUC values of the NBM for domain adaptation classifier learned using all the k-mers (denoted by *k-mersAll*) are also presented with italic font.

Source→Target	Unlabeled Data	20%	40%	60%	80%
CE→DM	b-mers	0.6466	0.6404	0.6346	0.639
	k-mers	0.4098	0.4898	0.4786	0.472
	k-mersAll	<i>0.5236</i>	<i>0.5314</i>	<i>0.5182</i>	<i>0.5392</i>
DM→CE	b-mers	0.7088	0.7044	0.6954	0.6966
	k-mers	0.5778	0.648	0.6544	0.6498
	k-mersAll	<i>0.6096</i>	<i>0.609</i>	<i>0.5942</i>	<i>0.5854</i>
gNeg→gPos	b-mers	0.9284	0.9208	0.918	0.9078
	k-mers	0.7806	0.7116	0.7104	0.7038
	k-mersAll	<i>0.9076</i>	<i>0.8804</i>	<i>0.8848</i>	<i>0.8528</i>
gPos→gNeg	b-mers	0.9378	0.9322	0.9188	0.8952
	k-mers	0.8952	0.9014	0.9064	0.912
	k-mersAll	<i>0.88</i>	<i>0.8568</i>	<i>0.8436</i>	<i>0.7896</i>
nonPlant→plant	c-mers	0.8604	0.7952	0.7742	0.7598
	k-mers	0.7558	0.716	0.7138	0.7112
	k-mersAll	<i>0.7294</i>	<i>0.7202</i>	<i>0.7182</i>	<i>0.7236</i>
plant→nonPlant	c-mers	0.8392	0.8332	0.803	0.7284
	k-mers	0.7894	0.7684	0.7478	0.7506
	k-mersAll	<i>0.682</i>	<i>0.6242</i>	<i>0.6216</i>	<i>0.6168</i>

more closely related than plant/non-plant or CE/DM organisms.

Chapter 7

Community Detection Approach: Experiments and Results

Section 7.1 of this chapter lists the set of more specific research questions related to the CDA-based approach addressed in this work. Section 7.2 provides details regarding default parameters and experiments performed to address research questions. Finally, Section 7.3 includes discussion and analysis of results.

7.1 Research Questions

1. *How does the number of c -mers compare to the number of k -mers?*

Features constructed using the CDA-based approach satisfy the ZOMOPS constraint (Section 4.2); therefore, dimensionality of the set of c -mers is expected to be very small compared to dimensionality of the set of k -mers. To investigate this property, the total number of c -mers was compared to the total number of k -mers in a supervised learning scenario in which all available data was assumed to be labeled.

2. *When feature selection is used in supervised learning, will c -mers or k -mers result in more informative features?*

Similar to b -mers in Section 6.1, in order to evaluate top features from c -mers and k -mers, feature selection was applied to select the same number of features from both feature sets. Classifiers were then learned from each subset and their performances were compared by varying the number of selected features.

3. *How does the predictive power of c -mers compare to the predictive power of k -mers in an SSL scenario?*

In SSL, feature selection techniques may not accurately capture feature-class dependencies, thereby misleading the classifier. Because dimensionality of c -mers is much smaller than dimensionality of k -mers and considering the unsupervised nature of the CDA-based approach, the predictive power of c -mers was investigated in an SSL scenario.

4. *How does the performance of c -mers compare to the performance of k -mers in the domain adaptation scenario for various amounts of target unlabeled data?*

For domain adaptation, as discussed in Section 5.3, data from two domains was made available: source domain and target domain. The predictive power of c -mers was investigated in a domain adaptation learning scenario when the amount of available labeled data from target domain is small.

5. *How do results vary with amounts of unlabeled data?*

Intuitively, for the same amount of labeled data (target labeled data in domain adaptation scenario), increasing the amount of unlabeled data (target unlabeled data in domain adaptation scenario) may result in increased performance; therefore, the amounts of unlabeled data was varied and variation in the performance of the classifiers in semi-supervised and domain adaptation learning scenarios was observed.

6. *How does the the number of top motifs selected, t , affect the performance of the supervised learning classifier?*

Each motif obtained from the CDA-based approach is associated with a set of l -mers. The total number of features (c -mers) generated increase with the number of motifs selected from each of the R samples. Classifier performance, in general, increases with the number of features. However, the motifs were sorted based on significance score for nucleotide sequences, and substitution score for protein sequences. Therefore, as t is increased, less significant motifs are added to the feature set, potentially adversely affecting classifier performance. Therefore, performance of the classifier learned from top t motifs, where t is varied, should be investigated. Performance of the classifiers are expected to initially increase and then possibly decrease with an increase in the value of t .

7. *What is the effect of the number of samples, R , and sample size, S , on the performance of the classifier in a supervised learning scenario?*

In supervised learning scenario, what combination of R and S among (a) (large R , large S), (b) (relatively small R , small S), and (c) (relatively small R , large S) yields the best results? Combination (b) is faster than the other two combinations. Ideally, (b) should give the best feature set.

8. *How does the Hamming distance threshold, x , used to construct the network, and maximum allowed mismatches, d , affect the predictive power of c -mers in classifying nucleotide sequences in a supervised learning scenario?*

In nucleotide sequences, varying x affects network size and density of the network, thereby affecting identified communities. Alternatively, varying d affects community size. In order to understand this behavior, x and d were varied and the corresponding performance of the final supervised learning classifier was recorded.

9. *For protein sequences, how does the threshold on the substitution score, s , affect the performance of the classifier in a supervised learning scenario?*

Varying the threshold s affects the number of edges in the network, thereby affecting communities obtained from the community detection algorithm. Specifically, use of extreme values of s creates a very dense network (small s) or sparse network (large s), potentially misleading the community detection algorithm. In order to study this behavior, the threshold, s , was varied and the performance of the supervised learning classifier was recorded.

7.2 Parameters and Experiments

7.2.1 Default Parameters

- Length of features: Feature length in the CDA-based approach is controlled by the parameter l . In order to generate variable length c -mers, parameter l was varied accordingly. Following values of l for nucleotide and protein sequences were chosen:
 - Nucleotide sequences: 6, 7, and 8
 - Protein sequences: 2, 3, and 4
- Total number of sequences in the dataset was denoted by N , and it varies for each dataset .
- Maximum number of mismatches allowed when filtering motif instances from identified communities for nucleotide sequences, $d = 1$
- Minimum substitution score threshold allowed in filtering motif instances from identified communities for protein sequences, $s = 15$
- Maximum Hamming distance between two nodes while constructing the initial network for nucleotide sequences, $x = 2$
- Minimum size of the community, $q = 5$

- Number of top motifs selected based on the associated score, $t = 10$
- Number of samples, $R = 50$
- Number of sequences per sample, $S = 10$

Based on the type of experiment conducted, to evaluate a particular parameter, we vary it and fix the remaining parameters accordingly.

7.2.2 Experiments

This section describes the set of experiments conducted to address research questions corresponding to the CDA-based approach discussed in Section 7.1. In order to generate c -mers, R samples were randomly selected with each sample containing S sequences, from the total available train sequences. As a result, the total number of sequences used to generate c -mers might be smaller than the total number of train sequences. Let N_{RS} be the set of unique sequences obtained from R samples.

Supervised learning: Similar to b -mers, 5-fold cross-validation was used to evaluate c -mers, as described in Section 5.3, and compared the predictive power of c -mers to k -mers.

- **Generating c -mers:** To generate c -mers, the CDA-based approach was invoked with length of motif (l), minimum community size (q), number of samples (R), and sample size (S) as parameters. To construct the network and extract motifs, maximum number of mismatches between the motif consensus and motif instance (d) and maximum Hamming distance to construct the network (x) were used for nucleotide sequences; the minimum substitution score (s) was used for protein sequences. The CDA-based approach returned the set of c -mers, and the number of c -mers were denoted by D_{c-mers} .
- **Generating k -mers:** In order to perform a fair comparison with the proposed CDA-based approach, the sliding window-based approach (Section 4) with $k = l$ was invoked

on the reduced set of train sequences (N_{RS}) in the process of generating k -mers. The resulting number of k -mers was denoted by $D_{k\text{-mers}}$.

In order to generate features of variable length, the above process was repeated for various values of l . A comparison of $D_{c\text{-mers}}$ with $D_{k\text{-mers}}$ addressed the first question from Section 7.1.

To answer the second question from Section 7.1, feature selection on labeled data represented using both c -mers (referred to as $Lab_{c\text{-mers}}$) and k -mers (referred to as $Lab_{k\text{-mers}}$) was applied separately to select top f features, similar to evaluation of b -mers. Let $f_{c\text{-mers}}$ be top f features selected from $Lab_{c\text{-mers}}$, and $f_{k\text{-mers}}$ be top f features selected from $Lab_{k\text{-mers}}$. In these experiments, f was varied from 50 to 1500. In order to evaluate resulting sets of features, train and test data were represented using $f_{c\text{-mers}}$ and $f_{k\text{-mers}}$, respectively. Two classifiers were then learned for the two feature sets $f_{c\text{-mers}}$ and $f_{k\text{-mers}}$ from corresponding train data; test data was then predicted using the two classifiers. Similar to the behavior of b -mers as described in Section 6.2.2, if c -mers contain most informative features, performance of the classifier learned from $f_{c\text{-mers}}$ should be better than the performance of the classifier learned from $f_{k\text{-mers}}$.

Semi-supervised scenario: Because the CDA-based approach is unsupervised, both labeled and unlabeled data were used in the process of generating c -mers and k -mers. Similar to the supervised setting, k -mers were generated from the set of N_{RS} sequences. To evaluate c -mers in an SSL framework, similar to b -mers, feature selection was used on labeled data represented using k -mers in order to select top $D_{c\text{-mers}}$ number of features. Labeled, unlabeled, and test data were then represented using c -mers and reduced set of k -mers. Encoded labeled and unlabeled data were further used to learn SSL classifiers, which were used to predict corresponding test data. In order to understand the predictive power of features in a semi-supervised scenario, the amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%, thereby answering the third and fifth questions from Section 7.1.

Domain adaptation scenario: Because class labels were not taken into account, all train data (labeled and unlabeled) from target domain was used to generate c -mers as described in Section 4.1.2. To generate k -mers, the sliding window-based approach was invoked with N_{RS} sequences. Similar to experimental setup for semi-supervised learning scenario, feature selection was used on target labeled data represented using k -mers in order to select top $D_{c\text{-mers}}$ features. Source labeled, target labeled, target unlabeled and target test data were then represented using c -mers and k -mers (reduced to dimensionality of $D_{c\text{-mers}}$). Domain adaptation classifier (described in Section 2.2.1) was trained using source labeled, target labeled, and target unlabeled data (represented using c -mers and reduced set of k -mers). The classifier was then used to predict target test data, and AUC was recorded. The amount of target labeled data was fixed to 20% and target unlabeled data was varied from 20% to 80%, thereby addressing the fourth and fifth questions from Section 6.1.

In order to investigate the effect of the number of top motifs selected, t (to address the sixth question from Section 7.1), default values for parameters $\{l, (\text{nucleotide sequences: } d, x; \text{ protein sequences: } s), q, R, S\}$ were used and t was varied from 1 to 50. For each value of t , performance of the classifier learned from the corresponding set of c -mers was recorded. This experiment was conducted in a supervised learning scenario and all c -mers (as opposed to selecting top f features) were used to represent train and test sequences. The seventh question was addressed by fixing parameters $\{l, (\text{nucleotide sequences: } d, x; \text{ protein sequences: } s), q, t\}$ and varying the number of samples, R , from 10 to 100, as well as sample size, S , from 10 to 50, respectively, in a supervised learning scenario. Finally, for eight and ninth questions, the parameters $\{l, q, t, R, S\}$ were fixed, and d and x were varied for nucleotide sequences, and s was varied for protein sequences in a supervised learning scenario.

Table 7.1: Comparison of the number of features generated using c -mers and k -mers for the six datasets used, averaged over 5 folds.

Dataset	c -mers	k -mers
CE	4700	67167
DM	7076	75371
gPos	1976	74203
gNeg	1823	83060
plant	1684	77091
nonPlant	1751	102815

7.3 Results

7.3.1 Dimensionality Comparison

In order to compare the number of features generated using the proposed approach with default parameters (c -mers) to the total number of k -mers, the count of the total number of features for each of the two feature sets were reported in Table 7.1. Similar to the experiment that compared dimensionality of b -mers to k -mers (Section 6.3.1), this experiment was conducted in a supervised learning scenario. All the unique sequences obtained from R samples were used to generate k -mers. As shown in Table 7.1, the total number of c -mers was significantly smaller than the total number of k -mers. The remainder of the experiments focused on evaluating the predictive power of c -mers compared to k -mers, and understanding the behavior of the CDA-based approach with different parameters.

7.3.2 Supervised Scenario: c -mers versus k -mers

This section evaluates the predictive power of c -mers in a supervised learning scenario. As discussed in Section 7.2, feature selection was used using all available labeled data in order to select top f features (from c -mers and k -mers, respectively) that were further used to represent sequences. From each representation, NBM classifiers (Section 2.2.1) using all available labeled (train) data were learned, and the classifiers were used to predict test data.

Table 7.2: Variation of the performance with the number of c -mers and k -mers features selected for NBM classifier in supervised learning scenario. Table includes AUC values for CE, DM, gPos, gNeg, plant, and nonPlant datasets. For each dataset, maximum AUC between b -mers and k -mers for each variation of the number of features selected is reported in **bold** font. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k -mers, AUC values of the NBM classifier learned using all the k -mers (denoted by k -mersAll) are also presented with italic font.

Index	1	2	3	4	5	6	7	8
f	50	100	150	200	250	500	1000	1500
CE(c -mers)	0.681	0.715	0.731	0.745	0.757	0.776	0.781	0.782
CE(k -mers)	0.558	0.577	0.57	0.586	0.588	0.683	0.694	0.729
CE(k -mersAll)	<i>0.773</i>							
DM(c -mers)	0.443	0.675	0.717	0.709	0.719	0.711	0.729	0.727
DM(k -mers)	0.411	0.399	0.407	0.387	0.388	0.398	0.466	0.539
DM(k -mersAll)	<i>0.568</i>							
gPos(c -mers)	0.746	0.792	0.812	0.824	0.826	0.876	0.912	0.923
gPos(k -mers)	0.649	0.698	0.736	0.77	0.786	0.838	0.86	0.87
gPos(k -mersAll)	<i>0.95</i>							
gNeg(c -mers)	0.77	0.841	0.857	0.869	0.876	0.904	0.918	0.925
gNeg(k -mers)	0.625	0.667	0.698	0.711	0.744	0.809	0.884	0.9
gNeg(k -mersAll)	<i>0.955</i>							
plant(c -mers)	0.65	0.707	0.737	0.743	0.748	0.784	0.823	0.836
plant(k -mers)	0.536	0.566	0.578	0.594	0.598	0.651	0.723	0.757
plant(k -mersAll)	<i>0.843</i>							
nonPlant(c -mers)	0.707	0.759	0.775	0.787	0.793	0.807	0.825	0.833
nonPlant(k -mers)	0.655	0.7	0.734	0.759	0.788	0.829	0.856	0.871
nonPlant(k -mersAll)	<i>0.868</i>							

In order to address the second question from Section 7.1, the performance of classifiers learned from c -mers was compared to the performance of classifiers learned from k -mers, while varying the number of features, f . f was varied as $\{50, 100, 150, 200, 250, 500, 1000, 1500\}$.

AUC values of NBM classifiers learned for various number of features, f , for c -mers and k -mers were reported in Table 7.2. For each dataset, maximum AUC between c -mers and k -mers for each variation of the number of features selected is reported in **bold** font. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k -mers, AUC values of the NBM classifier learned using all the k -mers are also reported in Table 6.2, referred to as k -mersAll (*italicized*). To better understand

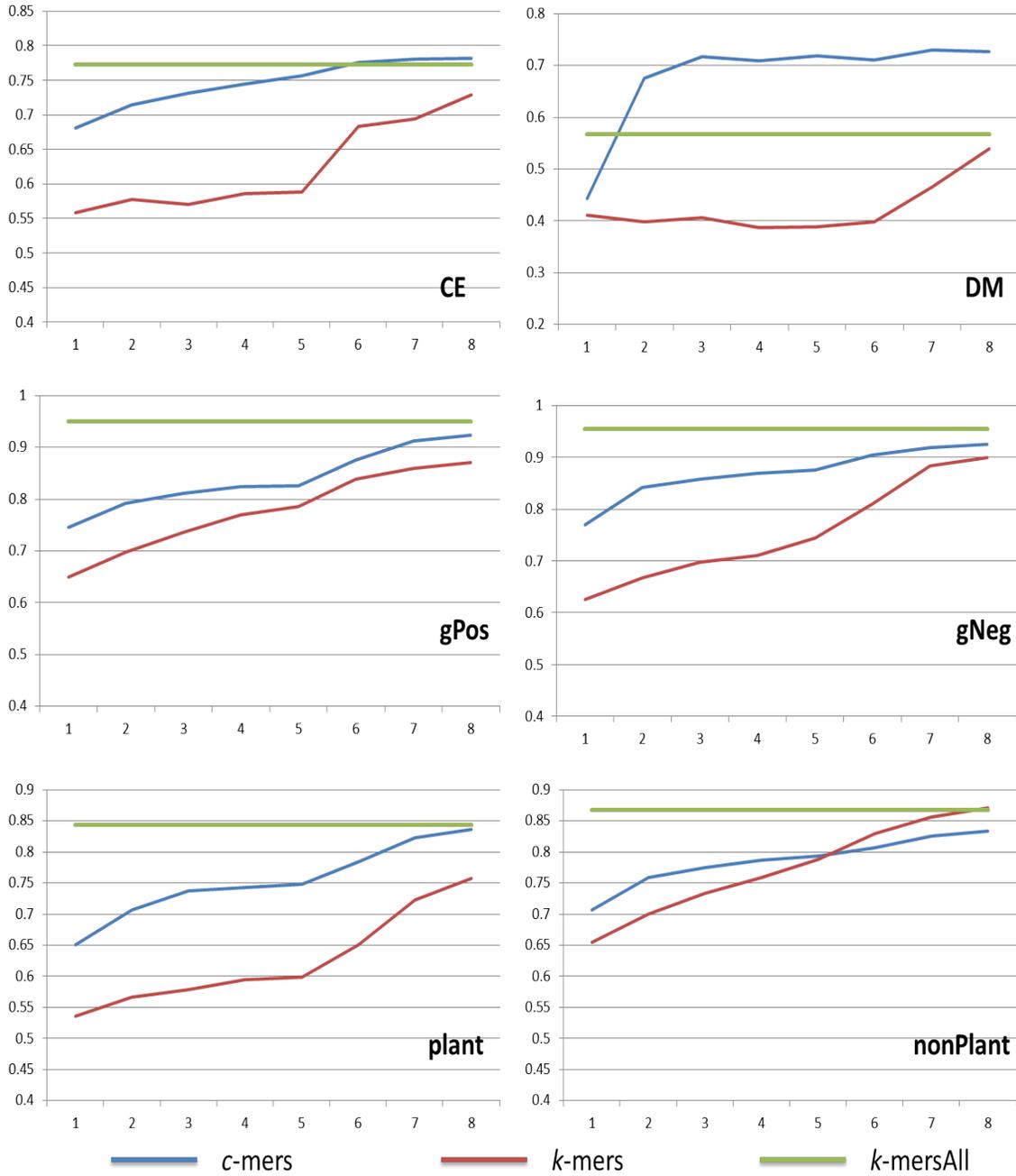


Figure 7.1: Variation of the performance of NBM classifier with the number of *c*-mers and *k*-mers features selected in supervised learning scenario is shown in the following graphs. Besides these, the performance of NBM classifier when all the *k*-mers are used as features is also shown in the graphs (*k*-mersAll). In each graph, *x*-axis represents the number of features selected from each feature set, *f*, which is varied from 50 to 1500. *y*-axis represents the AUC values of NBM classifier learned from the respective feature set. See Table 7.2 for the numbers shown on *x*-axis and the corresponding number of features selected, *f*.

the behavior, results are also plotted in Figure 6.1. The observation was made that c -mers outperformed k -mers in 45 out of 48 experiments (**bold** in Table 7.2). The remaining three experiments corresponded to nonPlant dataset. As discussed in Section 6.3.2, the nonPlant data was skewed to the third class (nuclear+cytosolic) as opposed to other datasets. Therefore, the hypothesis was made that for nonPlant dataset, most of the R samples potentially had more than half of the sequences belonging to the third class, possibly resulting in motifs that were biased to the third class. As a result, c -mers primarily captured information corresponding to the third class, thereby explaining why k -mers were better than c -mers for nonPlant dataset.

Furthermore, as shown in Figure 7.1, similar to the BWT-based approach, for nucleotide sequences, the classifier learned from c -mers outperformed the classifier learned from all k -mers, suggesting that the CDA-based approach successfully filtered uninformative k -mers. For protein sequences, it can be observed that the performance of the classifiers learned using b -mers is comparable to the performance of the classifiers learned for relatively larger number of features. This suggests that a comparable (or even better, in case of nucleotide sequences) performance is achieved with a small set of features.

Overall, results suggested that the proposed approach successfully retained informative features while reducing dimensionality to a large extent. Although the set of k -mers included the set of c -mers, when feature selection was applied, k -mers did not return features that were as informative as the ones obtained from c -mers. Subsequences responsible for a particular biological problem are believed to possibly span different sequences belonging to the same class of problems, with possible mismatches. The CDA-based approach was used to primarily identify subsequences that occurred among various sequences with certain mismatches, for nucleotide sequences, or subsequences that were similar to each other based on substitution score, for protein sequences. Therefore, c -mers contained a small set of highly informative features, and when feature selection was applied on top of this set of features, feature class dependency scores were computed more precisely because all feature

class dependency scores were normalized by maximum Shannon Entropy of all features, which, in the case of c -mers, is limited to only informative feature variations.

7.3.3 Semi-supervised Scenario: c -mers *versus* k -mers

As discussed in Section 7.2, the available train data was split into various combinations of labeled and unlabeled data. For each combination of labeled and unlabeled data, ST and CT classifiers were learned for all the six datasets (two nucleotide sequence and four protein sequence datasets) represented using c -mers and k -mers; k -mers were reduced to the number of c -mers using feature selection. In order to address the fifth question from Section 7.1, parameters $\{l, d, x, q, t, R, S\}$ were fixed to their default values, the amount of labeled data was fixed to 10%, and the amount of unlabeled data was varied as follows: {20%, 30%, 40%, 60%, 80%, 90%}. Besides using the reduced number of k -mers, to get an estimate of the best AUC that can be achieved using k -mers, AUC values of ST and CT classifiers learned using all the k -mers are also reported in Table 6.2, referred to as k -mersAll (*italicized*).

Table 7.3 reports AUC values of the set of experiments conducted in an SSL scenario. For each dataset, maximum AUC between c -mers and k -mers for each variation of the amount of unlabeled data is reported in **bold** font. To better understand the behavior, results are also plotted in Figures 7.2 and 7.3 for ST and CT, respectively. Results showed that c -mers outperformed k -mers in 44 out of 48 cases with ST, and 45 out of 48 cases with CT, suggesting that, with small amounts of labeled data, feature selection was unable to accurately capture feature-class dependencies, thereby selecting a set of uninformative features (selected features from k -mers). Furthermore, similar to the BWT-based approach, based on the plots from Figures 7.2 and 7.3, in majority of the cases, the classifier learned from c -mers outperformed the classifier learned from all k -mers in SSL scenario.

Ideally, when the amount of unlabeled data is increased, the performance of any SSL classifier should improve because the more the unlabeled data, the better is the classifier trained at each iteration. However, additional unlabeled data can occasionally result in

misabeled data being used for training, thereby degrading classifier performance. Therefore, the performance may initially increase and then decrease with increased amount of unlabeled data, as observed in 8 out of 12 possible combinations (ST: CE, DM, gPos, plant; CT: DM, gPos, plant, nonPlant). With sequences represented using c -mers, ST and CT had comparable performances on all datasets, and for each dataset, maximum AUC was obtained when CT was used with c -mers, suggesting that CT used the available unlabeled data more efficiently compared to ST.

7.3.4 Domain Adaptation Scenario: c -mers vs k -mers

Similar to the evaluation of b -mers in the domain adaptation scenario in Table 6.4, Table 7.4 shows AUC values of domain adaptation algorithm described in Section 2.2.1 learned using six source and target configurations, using varying amount of target unlabeled data (table columns) and c -mers or k -mers as features (table rows). For each comparison, the largest AUC value (between c -mers and k -mers) is highlighted with bold font.

In 23 out of 24 cases, the classifier had higher AUC when using c -mer features as compared to using k -mer features, suggesting that c -mers, by themselves, capture informative knowledge than k -mers, in order to predict data from the target domain.

7.3.5 Varying the Number of Motifs

Increasing the number of motifs selected, t , adds more motifs with less significant score, potentially affecting classifier performance. However, for very small t , the total number of l -mers obtained from the selected motifs might not be sufficient to carry enough information required to learn a good classifier. Therefore, classifier performance is expected to initially increase with t and later decrease, as observed for DM, with results shown in Table 7.5 and the variation of AUC values with the number of motifs selected t , was also plotted in Figure 7.5. For all other datasets, no decrease in the performance of the classifier was observed with increased t . However, the rate of increase was very small for larger values of t ,

Table 7.3: Variation of the performance with the amount of unlabeled data used with ST and CT classifiers, learned using c-mers and k-mers (reduced to the number of c-mers), respectively. Table includes AUC values for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, and the amount of unlabeled data was varied from 20% to 90%. For each dataset, maximum AUC between c-mers and k-mers for each variation of the amount of unlabeled data is reported in **bold font**. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k-mers, AUC values of the ST and CT classifiers learned using all the k-mers (denoted by *k-mersAll*) are also presented with *italic font*.

Self-training								
Index	1	2	3	4	5	6	7	8
Unlab	20%	30%	40%	50%	60%	70%	80%	90%
CE(<i>c</i> -mers)	0.653	0.638	0.646	0.632	0.652	0.658	0.613	0.655
CE(<i>k</i> -mers)	0.584	0.602	0.574	0.588	0.645	0.641	0.645	0.651
CE(<i>k</i> -mersAll)	<i>0.556</i>	<i>0.557</i>	<i>0.551</i>	<i>0.554</i>	<i>0.562</i>	<i>0.559</i>	<i>0.559</i>	<i>0.561</i>
DM(<i>c</i> -mers)	0.582	0.572	0.575	0.553	0.566	0.606	0.595	0.574
DM(<i>k</i> -mers)	0.527	0.524	0.532	0.532	0.53	0.534	0.53	0.534
DM(<i>k</i> -mersAll)	<i>0.517</i>	<i>0.517</i>	<i>0.516</i>	<i>0.517</i>	<i>0.516</i>	<i>0.516</i>	<i>0.517</i>	<i>0.516</i>
gPos(<i>c</i> -mers)	0.774	0.781	0.744	0.78	0.741	0.74	0.768	0.733
gPos(<i>k</i> -mers)	0.636	0.637	0.663	0.675	0.654	0.676	0.687	0.693
gPos(<i>k</i> -mersAll)	<i>0.722</i>	<i>0.708</i>	<i>0.702</i>	<i>0.708</i>	<i>0.676</i>	<i>0.671</i>	<i>0.691</i>	<i>0.663</i>
gNeg(<i>c</i> -mers)	0.844	0.833	0.841	0.826	0.835	0.834	0.819	0.834
gNeg(<i>k</i> -mers)	0.842	0.84	0.836	0.836	0.833	0.833	0.829	0.762
gNeg(<i>k</i> -mersAll)	<i>0.74</i>	<i>0.737</i>	<i>0.734</i>	<i>0.731</i>	<i>0.729</i>	<i>0.729</i>	<i>0.732</i>	<i>0.728</i>
plant(<i>c</i> -mers)	0.681	0.673	0.689	0.671	0.671	0.654	0.666	0.666
plant(<i>k</i> -mers)	0.608	0.616	0.61	0.609	0.609	0.613	0.606	0.604
plant(<i>k</i> -mersAll)	<i>0.616</i>	<i>0.614</i>	<i>0.604</i>	<i>0.606</i>	<i>0.601</i>	<i>0.602</i>	<i>0.602</i>	<i>0.601</i>
nonPlant(<i>c</i> -mers)	0.791	0.779	0.781	0.77	0.786	0.769	0.776	0.725
nonPlant(<i>k</i> -mers)	0.748	0.752	0.75	0.749	0.753	0.752	0.746	0.751
nonPlant(<i>k</i> -mersAll)	<i>0.59</i>	<i>0.589</i>	<i>0.589</i>	<i>0.585</i>	<i>0.586</i>	<i>0.586</i>	<i>0.586</i>	<i>0.584</i>
Co-training								
Index	1	2	3	4	5	6	7	8
Unlab	20%	30%	40%	50%	60%	70%	80%	90%
CE(<i>c</i> -mers)	0.648	0.65	0.668	0.649	0.614	0.632	0.66	0.642
CE(<i>k</i> -mers)	0.584	0.6	0.572	0.576	0.625	0.61	0.592	0.597
CE(<i>k</i> -mersAll)	<i>0.554</i>	<i>0.553</i>	<i>0.552</i>	<i>0.55</i>	<i>0.556</i>	<i>0.564</i>	<i>0.558</i>	<i>0.553</i>
DM(<i>c</i> -mers)	0.563	0.579	0.589	0.569	0.582	0.592	0.624	0.601
DM(<i>k</i> -mers)	0.521	0.515	0.522	0.527	0.527	0.531	0.525	0.528
DM(<i>k</i> -mersAll)	<i>0.517</i>	<i>0.517</i>	<i>0.516</i>	<i>0.517</i>	<i>0.516</i>	<i>0.516</i>	<i>0.517</i>	<i>0.516</i>
gPos(<i>c</i> -mers)	0.78	0.807	0.785	0.778	0.785	0.773	0.766	0.786
gPos(<i>k</i> -mers)	0.629	0.632	0.631	0.63	0.628	0.642	0.647	0.65
gPos(<i>k</i> -mersAll)	<i>0.717</i>	<i>0.707</i>	<i>0.7</i>	<i>0.681</i>	<i>0.685</i>	<i>0.659</i>	<i>0.677</i>	<i>0.68</i>
gNeg(<i>c</i> -mers)	0.847	0.842	0.847	0.846	0.841	0.832	0.828	0.832
gNeg(<i>k</i> -mers)	0.84	0.839	0.837	0.844	0.836	0.842	0.845	0.76
gNeg(<i>k</i> -mersAll)	<i>0.74</i>	<i>0.737</i>	<i>0.734</i>	<i>0.733</i>	<i>0.733</i>	<i>0.729</i>	<i>0.731</i>	<i>0.729</i>
plant(<i>c</i> -mers)	0.686	0.69	0.688	0.676	0.681	0.663	0.681	0.673
plant(<i>k</i> -mers)	0.604	0.607	0.606	0.612	0.606	0.613	0.604	0.608
plant(<i>k</i> -mersAll)	<i>0.616</i>	<i>0.614</i>	<i>0.605</i>	<i>0.606</i>	<i>0.601</i>	<i>0.602</i>	<i>0.602</i>	<i>0.601</i>
nonPlant(<i>c</i> -mers)	0.792	0.783	0.79	0.799	0.795	0.771	0.782	0.735
nonPlant(<i>k</i> -mers)	0.745	0.745	0.746	0.748	0.737	0.746	0.739	0.745
nonPlant(<i>k</i> -mersAll)	<i>0.59</i>	<i>0.589</i>	<i>0.588</i>	<i>0.585</i>	<i>0.586</i>	<i>0.585</i>	<i>0.586</i>	<i>0.585</i>

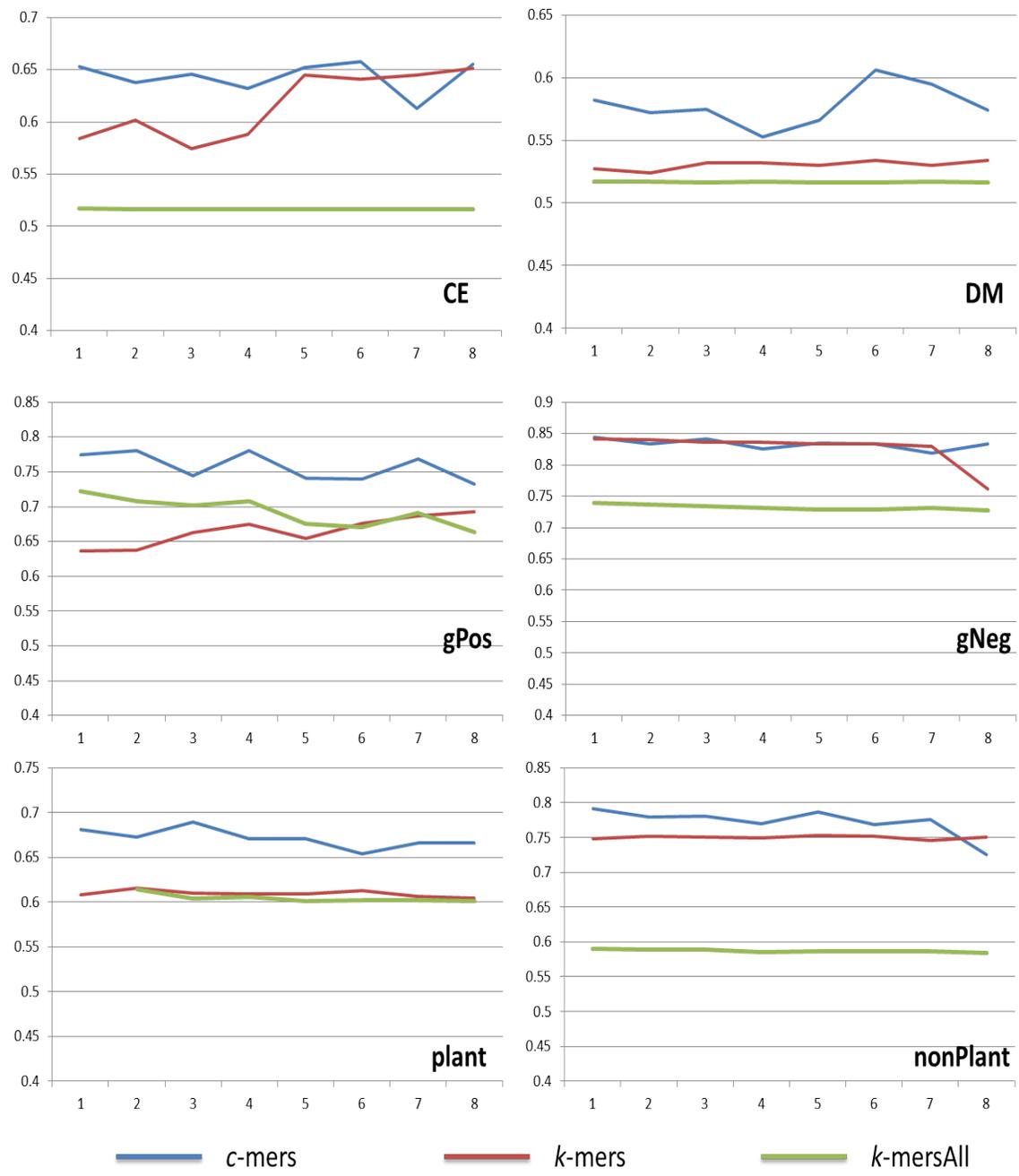


Figure 7.2: Variation of the performance with the amount of unlabeled data used with ST classifier, learned using c-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on x-axis). See Table 7.3 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

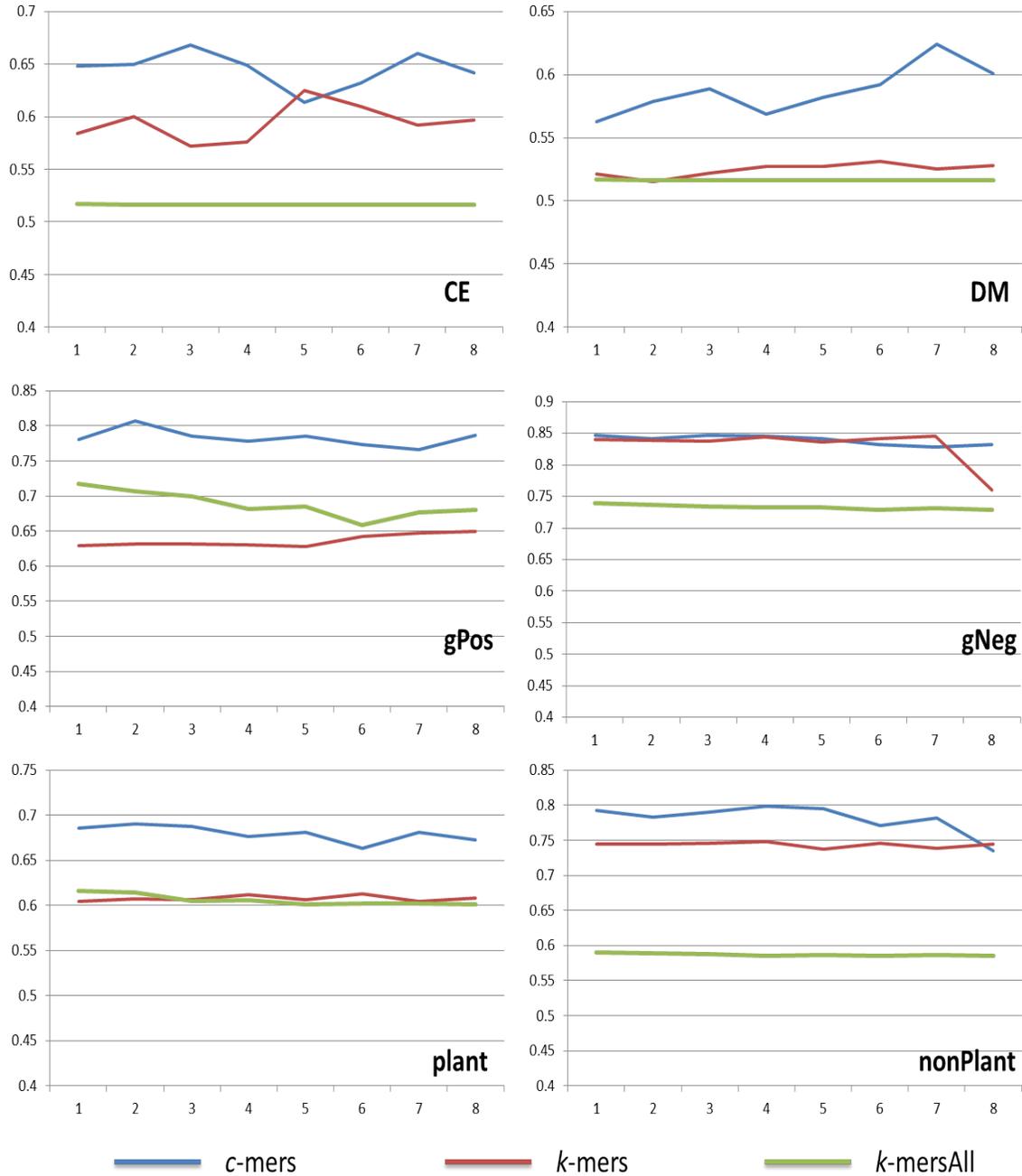


Figure 7.3: Variation of the performance with the amount of unlabeled data used with CT classifier, learned using c-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on x-axis). See Table 7.3 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

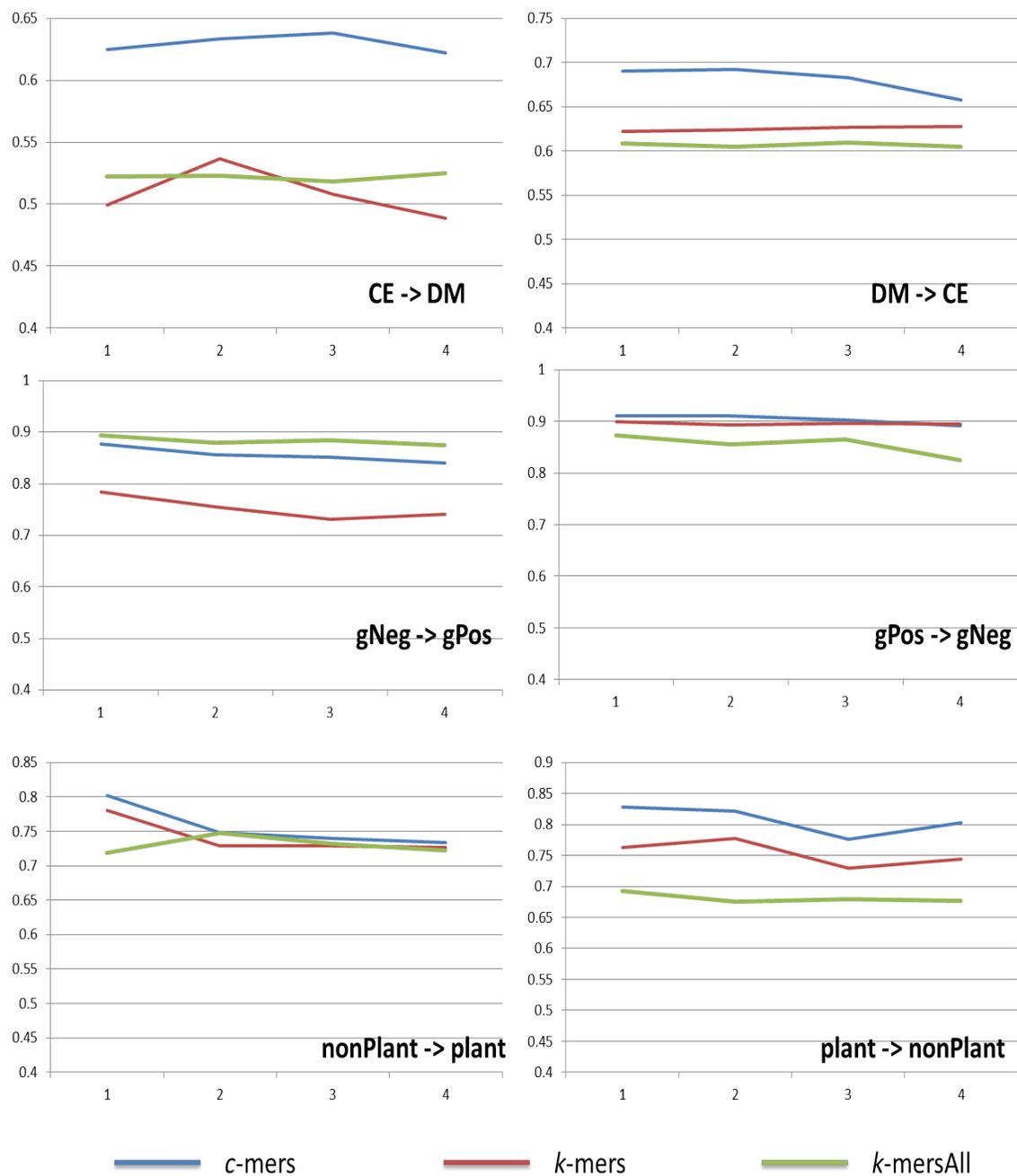


Figure 7.4: Variation of the performance with the amount of unlabeled data used with NBM for domain adaptation classifier, learned using c-mers, k-mers (reduced to the number of b-mers), and all k-mers (k-mersAll), respectively. Each graph plots the AUC values (on y-axis) for CE→DM, DM→CE, gPos→gNeg, gNeg→gPos, plant→nonPlant, and nonPlant→plant combinations of source→target datasets when the amount of labeled data was fixed to 20%, while the amount of unlabeled data was varied from 20% to 80% (on x-axis). See Table 7.4 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

Table 7.4: Variation of the performance with the amount of unlabeled data used with NBM for domain adaptation classifier, learned using c-mers and k-mers (reduced to the number of c-mers), respectively. Table includes AUC values for the following combinations of source and target datasets: $CE \rightarrow DM$, $DM \rightarrow CE$, $gPos \rightarrow gNeg$, $gNeg \rightarrow gPos$, $plant \rightarrow nonPlant$, and $nonPlant \rightarrow plant$. For each combination of the dataset, the amount of unlabeled data used and the largest AUC value (between c-mers and k-mers) is highlighted with **bold** font. For all experiments, the amount of labeled data is fixed to 20%. Besides using the reduced number of features, to get an estimate of the best AUC that can be achieved using k-mers, AUC values of the NBM for domain adaptation classifier learned using all the k-mers (denoted by *k-mersAll*) are also presented with *italic* font.

Source→Target	Unlabeled Data	20%	40%	60%	80%
CE→DM	c-mers	0.6248	0.6332	0.6378	0.6222
	k-mers	0.4996	0.5368	0.5078	0.4888
	k-mersAll	<i>0.5222</i>	<i>0.5228</i>	<i>0.5184</i>	<i>0.5248</i>
DM→CE	c-mers	0.69	0.692	0.6828	0.658
	k-mers	0.622	0.624	0.6272	0.6282
	k-mersAll	<i>0.6082</i>	<i>0.605</i>	<i>0.6096</i>	<i>0.605</i>
gNeg→gPos	c-mers	0.8766	0.8564	0.852	0.8394
	k-mers	0.7846	0.7546	0.7314	0.741
	k-mersAll	<i>0.894</i>	<i>0.88</i>	<i>0.8836</i>	<i>0.8748</i>
gPos→gNeg	c-mers	0.911	0.9104	0.902	0.8918
	k-mers	0.8986	0.893	0.896	0.895
	k-mersAll	<i>0.8722</i>	<i>0.8558</i>	<i>0.8644</i>	<i>0.8244</i>
nonPlant→plant	c-mers	0.802	0.7476	0.7394	0.7338
	k-mers	0.7804	0.7284	0.7284	0.7266
	k-mersAll	<i>0.7192</i>	<i>0.747</i>	<i>0.7314</i>	<i>0.722</i>
plant→nonPlant	c-mers	0.8286	0.821	0.7766	0.8034
	k-mers	0.763	0.7774	0.73	0.7444
	k-mersAll	<i>0.6932</i>	<i>0.6754</i>	<i>0.6792</i>	<i>0.6772</i>

suggesting that increasing t may maintain or decrease the performance, thereby answering the sixth question from Section 7.1.

7.3.6 Varying the Number of Samples and Sample Size

In order to address the seventh question of Section 7.1, the parameters: $\{l, d, x, p, f, S\}$ were fixed and R was varied as $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ in a supervised learning scenario, as discussed in Section 7.2. Although parameters $\{l, d, x, p, f\}$ were assigned to their default values, parameter S was chosen to have different values: 10, 20,

Table 7.5: AUC values obtained with NBM classifiers learned in the supervised scenario, when top t motifs are used as features. The number of motifs, t , was varied from 1 to 50, and the performance was recorded for each value of t . In addition to performance, we also show the total number of c -mers obtained for a set of t motifs ($\#c$ -mers), averaged over five folds.

	CE		DM		gPos	
t	$\#c$ -mers	AUC	$\#c$ -mers	AUC	$\#c$ -mers	AUC
1	1078	0.716	1114	0.652	152	0.824
5	3144	0.769	4203	0.704	867	0.905
10	4700	0.786	7075	0.717	1976	0.925
15	5901	0.796	9370	0.719	3158	0.936
25	8143	0.805	12986	0.717	5506	0.944
35	10386	0.808	15876	0.714	7700	0.945
50	13052	0.81	19471	0.698	10522	0.948

	gNeg		plant		nonPlant	
t	$\#c$ -mers	AUC	$\#c$ -mers	AUC	$\#c$ -mers	AUC
1	147	0.789	145	0.691	146	0.704
5	805	0.901	800	0.792	765	0.797
10	1822	0.929	1684	0.837	1750	0.834
15	2936	0.939	2593	0.854	2888	0.857
25	5139	0.951	4361	0.868	5357	0.88
35	7237	0.955	6097	0.876	7884	0.892
50	9800	0.956	8444	0.875	11323	0.898

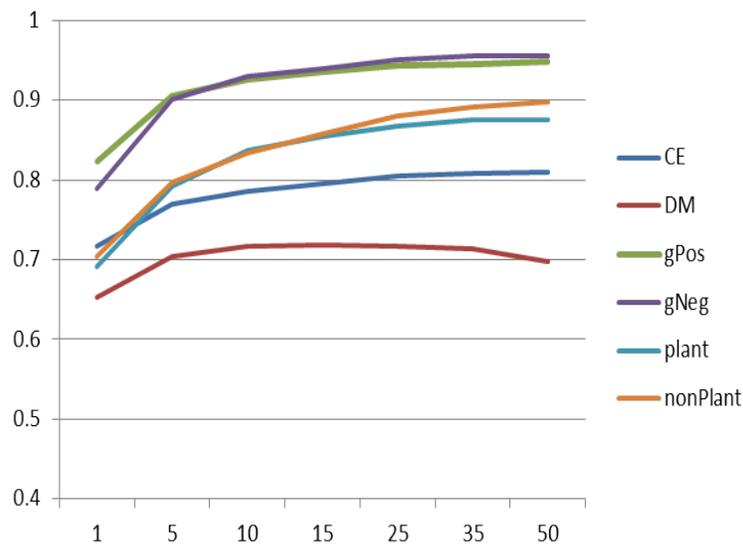


Figure 7.5: Variation of AUC values (y -axis) with CDA-based approach, when the number of top motifs selected, t , was varied from 1 to 50 (x -axis).

30, 40, 50. Figure 7.6 plots R (x -axis) versus AUC values (y -axis) for various values of S (curves in each plot) on all six datasets.

Varying the number of samples, R : Plots in Figure 7.6, show that for most values of S and datasets (except for $S=\{40, 50\}$ with DM), AUC values increased with an increase in R , suggesting that increasing the number of samples resulted in more informative motifs. However, the slope of the plots obtained for a particular value of S tends to 0 for larger values of R (specifically for R greater than 50), suggesting that performance does not rapidly increase with larger values of R . Therefore, in most cases, a good performance can be attained with a relatively small value of R (e.g., 50).

Varying the number of sequences per sample, S : As shown in the plots in Figure 7.6, for almost all datasets (except nonPlant dataset), AUC values of the classifiers learned from samples of 10 sequences ($S = 10$) outperformed the rest of the classifiers learned from larger samples in most cases, suggesting that the best performance was achieved for a smaller value of S . Because increasing S significantly increases the running time of the proposed approach, the best performance had very small running time compared to other sub-optimal performances.

Graphs corresponding to various values of S converged for large values of R in most cases, suggesting that the best performance was achieved for a relatively small value of R and a small value of S combination, thereby answering the seventh question from Section 7.1. As a result, features generated using the proposed approach were effective (in terms of performance) and efficient (in terms of running time) compared to TFBSGroup applied on a larger set of sequences.

7.3.7 Varying the Number of Mismatches and Hamming Distance

For nucleotide sequences, network construction and motif identification was affected by the maximum number of mismatches, d , and the Hamming distance, x . In order to understand the effect of these two parameters, one of them was fixed and the other was varied, thereby

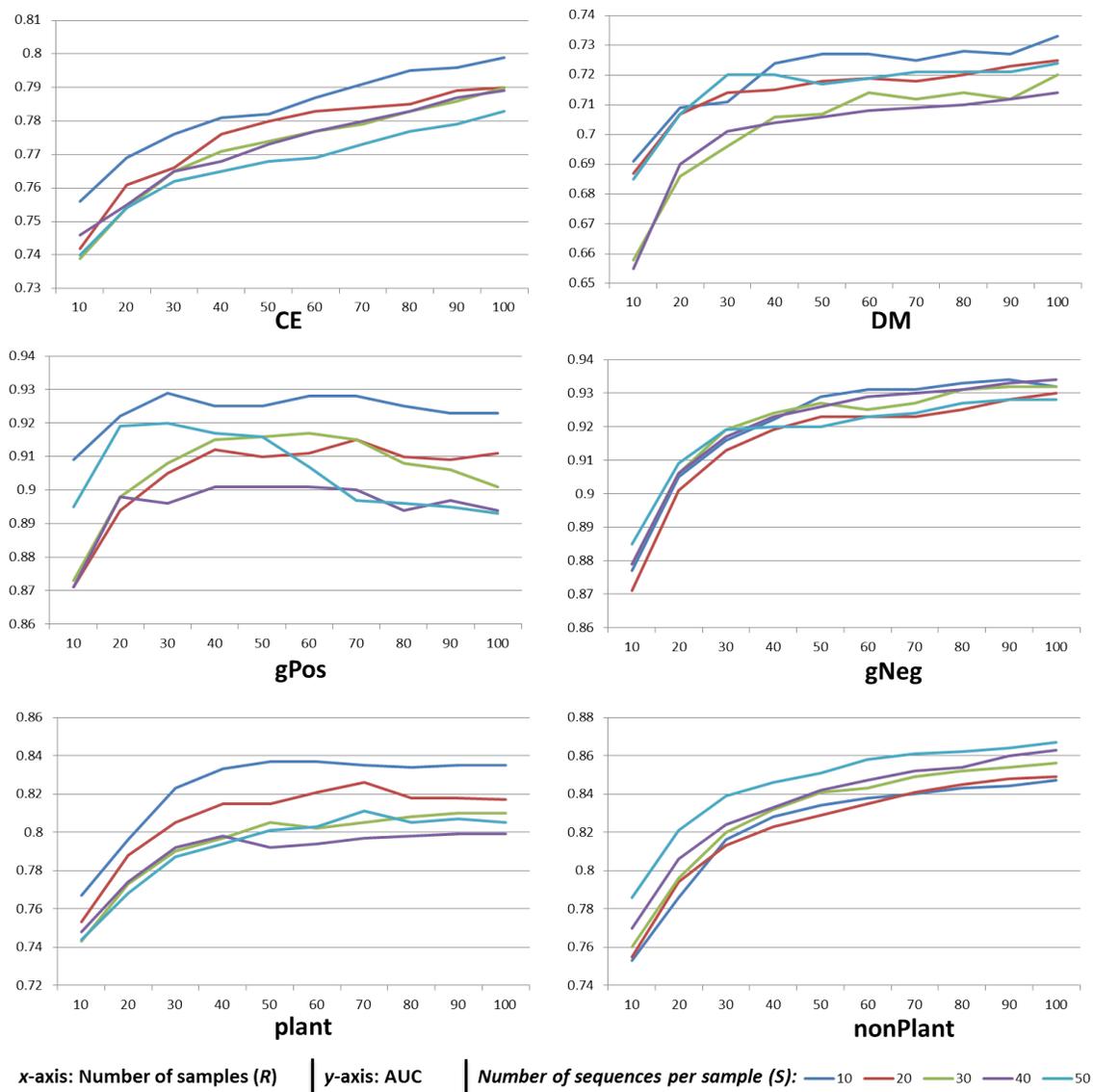


Figure 7.6: Variation of AUC values with CDA-based approach, when sample size, S , was varied from 10 to 50 with increments of 10, and number of samples, R , was varied from 10 to 100 (with increments of 10), on all six datasets (CE, DM, gPos, gNeg, plant, and nonPlant). The number of features used to represent sequences, f , was set to 2000.

Table 7.6: AUC values obtained with NBM classifiers learned in a supervised scenario, for nucleotide sequences when varying the maximum number of mismatches allowed, $d = \{1, 2\}$, while maximum Hamming distance considered when constructing the network, x , was fixed to 2. Each column corresponds to a particular (d, x) combination.

	CE		DM	
(d, x)	(1, 2)	(2, 2)	(1, 2)	(2, 2)
{6, 7, 8}	0.782	0.831	0.727	0.731

Table 7.7: AUC values obtained with NBM classifiers learned in a supervised scenario, for nucleotide sequences, when varying the Hamming distance considered when constructing the network, $x = \{1, 2\}$ by fixing the maximum number of mismatches allowed, $d = 1$. Each column corresponds to a particular (d, x) combination.

	CE		DM	
(d, x)	(1, 1)	(1, 2)	(1, 1)	(1, 2)
{6, 7, 8}	0.796	0.782	0.705	0.727

addressing the sixth question from Section 7.1. Tables 7.6 and 7.7 show AUC values obtained when classifiers were trained using respective c -mers with various values of d by fixing x and various values of x by fixing d .

As shown in Table 7.6, performance of the classifier generally increased with d , suggesting that motifs with relatively larger number of mismatches capture better information compared to motifs with smaller number of mismatches. Results from Table 7.7 suggest that a relatively large value of x helped the CDA-based approach generate informative features. However, increasing the value of x (which is not possible in this scenario because the maximum value of x is $2d$) is expected to significantly increase the network size, thereby adversely affecting the outcome of the CDA-based approach.

7.3.8 Varying the Substitution Score Threshold

Varying threshold s will affect the number of edges in the network, potentially affecting communities obtained from the community detection algorithm. For very small values of s , the network may be too sparse resulting in very small set of c -mers that may not be informative. Contrarily, for a very large value of s , network density may be very high, resulting

in a very large set of c -mers. As the dimensionality of c -mers increases, the probability of outliers in the resulting set of c -mers also increases. Therefore, when the threshold s increased, the performance of the classifier learned from the resulting set of c -mers was expected to initially increase and then decrease. From Table 7.8, this behavior was observed for plant and nonPlant datasets. For gNeg and gPos datasets, the performance decreased right from the beginning, suggesting that the optimal performance for these datasets might be obtained with a much smaller value of s ($s < 1$).

Table 7.8: *AUC values obtained with NBM classifiers learned in a supervised scenario, for protein motifs of lengths $\{2, 3, 4\}$, when varying the substitution score considered when constructing the network from 1:5:10:15 (shown on columns).*

s	1	5	10	15
gPos	0.882	0.888	0.913	0.925
gNeg	0.937	0.941	0.938	0.929
plant	0.833	0.841	0.84	0.837
nonPlant	0.871	0.872	0.868	0.834

Chapter 8

Hybrid Approach: Experiments and Results

Section 8.1 outlines the set of specific research questions related to the HBA approach, followed by the description of the set of default parameters and experiments in Section 8.2. Finally, results of the experiments are discussed in Section 8.3. This chapter evaluates the performance of all three proposed approaches by comparing them with two other feature representations:

- Feature hashing (referred to as r -mers from *r*andomly generated hash bins)
- Union of b -mers and c -mers (referred to as u -mers).

8.1 Research Questions

1. *In the supervised learning scenario, how does the predictive power of features constructed using the HBA approach compare to that of b -mers, c -mers, u -mers, and r -mers?*

The HBA approach is a hybrid approach that combines features generated using the

BWT-based approach with the CDA-based approach to generate BWT-based features with certain mismatches (h -mers). The set of u -mers, as opposed to the set of h -mers, is obtained by taking an union of b -mers and c -mers. While h -mers specifically target b -mers with mismatches, u -mers combine two feature sets that capture different types of information. In order to identify whether h -mers or u -mers perform better, experiments were conducted in supervised, semi-supervised, and domain adaptation learning scenarios. Furthermore, because h -mers include the entire set of b -mers and other subsequences close to the set of b -mers (obtained from the CDA-based approach), the predictive power of h -mers was expected to be better than that of b -mers and c -mers. The predictive power of features constructed using proposed approaches were also compared with feature hashing (r -mers).

2. *In the semi-supervised and domain adaptation learning scenarios, how does the predictive power of features constructed using the HBA approach compare to that of b -mers, c -mers, u -mers, and r -mers?*

Similar to supervised learning scenario, the predictive power of b -mers, c -mers, h -mers, and feature hashing (r -mers) were compared in an SSL and domain adaptation learning scenarios, with a varied amount of unlabeled data (in an SSL scenario) and target labeled and target unlabeled data (in domain adaptation scenario).

8.2 Parameters and Experiments

8.2.1 Default Parameters

The following default values were used to conduct experiments:

- Length of features: Feature length in the HBA approach was controlled by the parameter l . From the total set of variable length features obtained from the BWT-based approach, the features with length l were retained, and the CDA-based approach was

invoked with the same values of l ; l can be of variable length and, in this work, the following values of l were used for nucleotide and protein sequences:

- Nucleotide sequences: 6, 7, and 8
 - Protein sequences: 2, 3, and 4.
- Total number of sequences in the dataset, N
 - Maximum number of mismatches allowed when filtering motif instances from identified communities for nucleotide sequences, $d = 1$
 - Maximum Hamming distance to filter the initial set of l -mers (based on similarity with b -mers) for nucleotide sequences, $d_h = l/2$
 - Minimum substitution score threshold allowed in filtering motif instances from identified communities for protein sequences, $s = 15$
 - Minimum substitution score to filter the initial set of l -mers (based on similarity with b -mers) for protein sequences, $s_h = 0$
 - Maximum Hamming distance between two nodes while constructing the initial network for nucleotide sequences, $x = 2$
 - Minimum size of the community, $q = 5$
 - Number of top motifs selected based on the associated score, $t = 10$
 - Number of samples, $R = 50$
 - Number of sequences per sample, $S = 10$

8.2.2 Experiments

This section describes the experiments conducted to address research questions corresponding to HBA-based features discussed in Section 8.1. The HBA approach uses the CDA-based approach on top of features constructed using the BWT-based approach. Therefore, experimental setup of the HBA approach is similar to that of the CDA-based approach, in which R samples with each sample containing S number of sequences were randomly selected from the total available training sequences. Let N_{RS} be the set of unique sequences obtained from the R samples.

Supervised scenario: In order to perform a fair comparison of all feature sets, b -mers were also constructed from the total unique sequences (N_{RS}) obtained from R samples, as opposed to total training sequences. For feature hashing, no training data was necessary because hashing encodes a particular sequence, as opposed to using the set of sequences to construct features.

The process of generating b -mers, c -mers, h -mers, and r -mers is described below:

- **Generating b -mers:** In order to generate b -mers, as opposed to using all training data, to obtain a fair comparison with other approaches, the BWT-based approach (as described in Section 4.1.2) was invoked with the set of N_{RS} sequences, length of motif (l), and minimum number of occurrences of the motif in the original sequence (r) as input. The BWT-based approach returned the set of b -mers, and the number of b -mers was denoted by $D_{b\text{-mers}}$.
- **Generating c -mers:** In order to generate c -mers, the CDA-based approach was invoked with length of the motif (l), minimum community size (q), number of samples (R), and sample size (S) as parameters. Furthermore, for nucleotide sequences, the maximum number of mismatches between the motif consensus and motif instance (d), and maximum Hamming distance was used to construct the network. For protein sequences, the minimum substitution score (s) was used in the process of constructing

the network and refining the motif. However, contrary to selecting top t motifs, top t_b motifs were selected, where t_b was chosen at run time in such a way that the number of resulting c -mers were close to the number of b -mers in order to have a fair comparison. The increased set of c -mers was referred to as c_b -mers. In order to compare the CDA-based approach to the BWT-based approach and the HBA approach, c_b -mers, as opposed to c -mers were used to learn the classifiers. The number of c_b -mers is denoted by $D_{c_b\text{-mers}}$.

- **Generating h -mers:** In order to generate h -mers, the HBA approach was invoked with length of the motif (l), maximum number of mismatches between the motif consensus and motif instance (d), minimum community size (q), maximum Hamming distance to construct the network (x), number of samples (R), sample size (S) and BWT thresholds (d_h and s_h) as parameters. Similar to the process of generating c -mers, maximum number of mismatches between the motif consensus and the motif instance (d) and maximum Hamming distance were used to construct the network (x) for nucleotide sequences, and minimum substitution score (s) for protein sequences, in the process of refining the motif. The algorithm returned the set of h -mers, and the number of h -mers was denoted by $D_{h\text{-mers}}$.
- **Generating u -mers:** In order to generate u -mers, all possible unique features were selected from sets of b -mers and c -mers.
- **Generating r -mers:** In order to generate instances from sequences using feature hashing, each sequence was sent as an input to feature hashing along with the number of features, f , and lengths of the features. The input sequence was then transformed to an instance with f number of features (obtained as described in Section 2.2.2). This process was repeated for all the sequences from labeled and testing data.

In order to compare the predictive power of various feature sets in a supervised learning scenario, feature selection was applied on the labeled data (all the training data in supervised

scenario) represented using b -mers, c -mers, c_b -mers, h -mers, and u -mers, to select top f features from each set. For feature hashing, labeled and test sequences with the same value of f were sent as an input to feature hashing. f was varied from 50 to 1500 (specifically, 50, 100, 150, 200, 250, 500, 1000, 1500). Labeled and test sequences were then represented using the reduced set of b -mers, c -mers, c_b -mers, h -mers, u -mers, and r -mers. The NBM classifiers were then learned from resulting labeled instances, and the classifiers were used to predict corresponding test instances.

Semi-supervised scenario: Because class labels were not taken into account, all the N_{RS} sequences were used to generate b -mers, c -mers, c_b -mers, h -mers and u -mers. To generate r -mers, as described earlier, each sequence from labeled, unlabeled, and test were sent as an input to feature hashing algorithm, along with the number of features, f . Classification accuracy is believed to increase with dimensionality of the feature set (assuming that features do not act as noise); therefore, dimensionality of r -mers was reduced to the maximum dimensionality of b -mers, c -mers, u -mers, c_b -mers, and h -mers, which can be the maximum performance achieved when comparing to the proposed feature sets. Therefore,

$f = \text{Max} (|b\text{-mers}|, |c\text{-mers}|, |c_b\text{-mers}|, |h\text{-mers}|, |u\text{-mers}|)$; where $| |$ is the dimensionality of a particular feature set

The labeled, unlabeled and testing data were then represented using b -mers, c -mers, c_b -mers, h -mers, u -mers, and r -mers (with f features). Encoded labeled and unlabeled instances were then used to learn semi-supervised classifiers, which were further used to predict corresponding testing data. This process was repeated for all available feature sets and the performance of the classifiers was recorded. In order to understand the predictive power of features in the semi-supervised scenario, the amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%, thereby addressing the second question from Section 6.1.

Domain Adaptation scenario: Similar to semi-supervised setting, dimensionality of r -mers was reduced to the maximum dimensionality of b -mers, c -mers, u -mers, c_b -mers,

and h -mers, which can be the maximum performance achieved to compare to the proposed feature sets. Therefore,

$$f = \text{Max} (|b\text{-mers}|, |c_b\text{-mers}|, |h\text{-mers}|, |u\text{-mers}|); \text{ where } |c| \text{ is the dimensionality of } c$$

The target labeled, source labeled, target unlabeled and target test data were then represented using b -mers, c -mers, h -mers, u -mers, and r -mers (with f features). Encoded target labeled, source labeled and target unlabeled instances were then used to learn domain adaptation classifier, which was further used to predict corresponding target test data. This process was repeated for all available feature sets and the performance of the classifiers was recorded. In order to understand the predictive power of features in the domain adaptation scenario, the amount of labeled data was fixed to 20% and the amount of unlabeled data was varied from 20% to 80%, thereby addressing the second question from Section 6.1.

8.3 Results

8.3.1 Dimensionality Comparison

Table 8.3.1 presents dimensionality information of b -mers, c -mers, h -mers, and u -mers when generated using the same set of sequences (N_{RS}). Although the HBA approach combines the BWT and CDA-based approaches, it attempts to identify subsequences similar to b -mers, as opposed to all possible similar subsequences (c -mers). Therefore, c -mers include a larger set of feature, than those identified using the HBA approach (features present in h -mers but not in b -mers). Because u -mers combine b -mers and c -mers by taking all possible unique features, they have higher dimensionality as compared to other proposed feature sets.

8.3.2 Supervised Learning Scenario

Table 8.2 presents AUC values of predictions obtained from the NBM classifier on various datasets, which are represented using b -mers, c -mers, c_b -mers, h -mers, u -mers, and r -mers.

Table 8.1: Comparison of the number of features generated using *b*-mers, *c*-mers, *h*-mers, and *u*-mers for six datasets, averaged over 5 folds.

	<i>b</i> -mers	<i>c</i> -mers	<i>h</i> -mers	<i>u</i> -mers
CE	3110	4700	5323	6262
DM	3426	7076	6006	8650
gPos	3766	1976	4695	5320
gNeg	3681	1823	4075	5105
plant	2970	1684	3114	4319
nonPlant	5296	1751	6231	6629

The number of features in each set of features was varied from 50 to 1500, and the respective AUC values are reported. To better understand the behavior, results are also plotted in Figure 8.1. The following observations were derived from the results:

- ***b*-mers versus *c*-mers versus c_b -mers:** As shown in Table 8.2 and Figure 8.2, *b*-mers outperformed *c*-mers in 27 out of 48 cases (and the AUC values are comparable in remaining cases) for protein and nucleotide sequences, suggesting that the BWT-based approach captured better information than the CDA-based approach. However, when dimensionality of *c*-mers was varied to match the dimensionality of *b*-mers, (using c_b -mers) for nucleotide sequences, classifiers learned using *b*-mers outperformed classifiers learned using c_b -mers in majority of the cases (11 out of 16 cases). However, for protein sequences, classifiers learned using c_b -mers performed better than classifiers learned using *b*-mers (22 out of 32 cases).

One reason c_b -mers may perform better than *b*-mers with protein sequences could be explained by the average number of unique subsequences per motif obtained using the CDA-based approach. When top 10 motifs were selected from nucleotide and protein sequences, the total number of *c*-mers was smaller than the number of *b*-mers for protein sequences, while the number of *c*-mers was larger than the number of *b*-mers for nucleotide sequences, possibly due to the large search space for nucleotide sequences when compared to protein sequences (based on the note below). As a result, in order to generate c_b -mers, the number of selected motifs increased for protein sequences. Based

on observations from Section 7.3.5, classifier performance is believed to increase with the number of motifs to a certain extent and then we start to notice the decrease for DNA, but we see an increase for protein sequences. Furthermore, based on the nature of the problem, subsequences that are responsible for alternative splicing events might occur multiple times in the same sequence, as opposed to subsequences responsible for protein localization, which are expected to occur across different sequences.

Note: For nucleotide sequences, features of lengths 6, 7, 8 were selected, while for protein sequences, features of lengths 2, 3, 4 were selected. For BWT and CDA-based approaches, the number of features returned, decreases with an increase in the length of the features, because, the probability of a subsequence occurring multiple times in a given sequence decreases with length of the subsequence (captured using the BWT-based approach), and the probability of two subsequences having a particular similarity decreases with increased length of the subsequences (used to construct c -mers). Therefore, most b -mers/ c -mers/ c_b -mers were associated with smaller length features (e.g., 6 or 7 for nucleotide sequences and 2 or 3 for protein sequences). Furthermore, when used for constructing smaller length features, the total number of unique protein subsequences (maximum search space of protein subsequences for creating the network) was very small compared to the number of nucleotide subsequences used for the CDA-based approach. For example, there are a total of 400 unique protein subsequences of length 2, and 4096 unique subsequences of length 6 for nucleotide sequences.

- **b -mers versus h -mers:** As shown in Figure 8.3, for nucleotide sequences, b -mers and h -mers (b -mers with mismatches) demonstrated higher predictive power than other sets of features. When the number of selected features, f , was small, b -mers outperformed h -mers. For larger values of f , h -mers were better than b -mers for CE ($f = 500, 1000, 1500$) and were better ($f=100, 150, 200, 250$) / comparable ($f=500, 1000, 1500$) with b -mers in DM. As discussed in Section 4.3, BWT-based features do not take into account any mismatches. For small values of f , the effect of overlapping features was

negligible because highly informative non-overlapping features are included. However, as f increases, the addition of some redundancy (by including mismatches to the b -mers) may benefit the classifier, as opposed to including other non-overlapping features (present in b -mers).

For protein sequences, performances of classifiers learned using b -mers and h -mers were mostly comparable, with b -mers performing better than h -mers in 23 out of 36 cases. With a large alphabet size and small length of features, the addition of similar features was not as helpful as for nucleotide sequences.

- **c_b -mers versus h -mers:** h -mers are primarily derived from b -mers. Therefore, comparison of c_b -mers to h -mers, was expected to be similar to the comparison of c_b -mers to b -mers (as described earlier). As shown in Figure 8.4, the performance of classifiers learned using h -mers was better than the performance of the classifier learned using c -mers for nucleotide sequences (10 out of 16 cases). For protein sequences, c_b -mers produced better AUC values than h -mers in 23 out of 32 cases.
- **h -mers versus u -mers:** As shown in Figure 8.5, and based on observations made earlier in this section, for nucleotide sequences, b -mers performed better than c -mers. Therefore, h -mers, including b -mers and other subsequences similar to b -mers, gave better results in 34 out of 48 cases, compared to u -mers, which included b -mers and c -mers.
- **r -mers:** The proposed approaches outperformed hashing in majority of the cases. Hashing outperformed the proposed feature sets for two datasets (gPos and gNeg). Especially, the difference is remarkable for gPos dataset. The reason for this could be the dataset size of gPos. The total number of sequences in gPos dataset are 540, which is relatively small compared to other datasets, thereby reducing the ability of the proposed approaches to capture sufficient information corresponding to different classes.

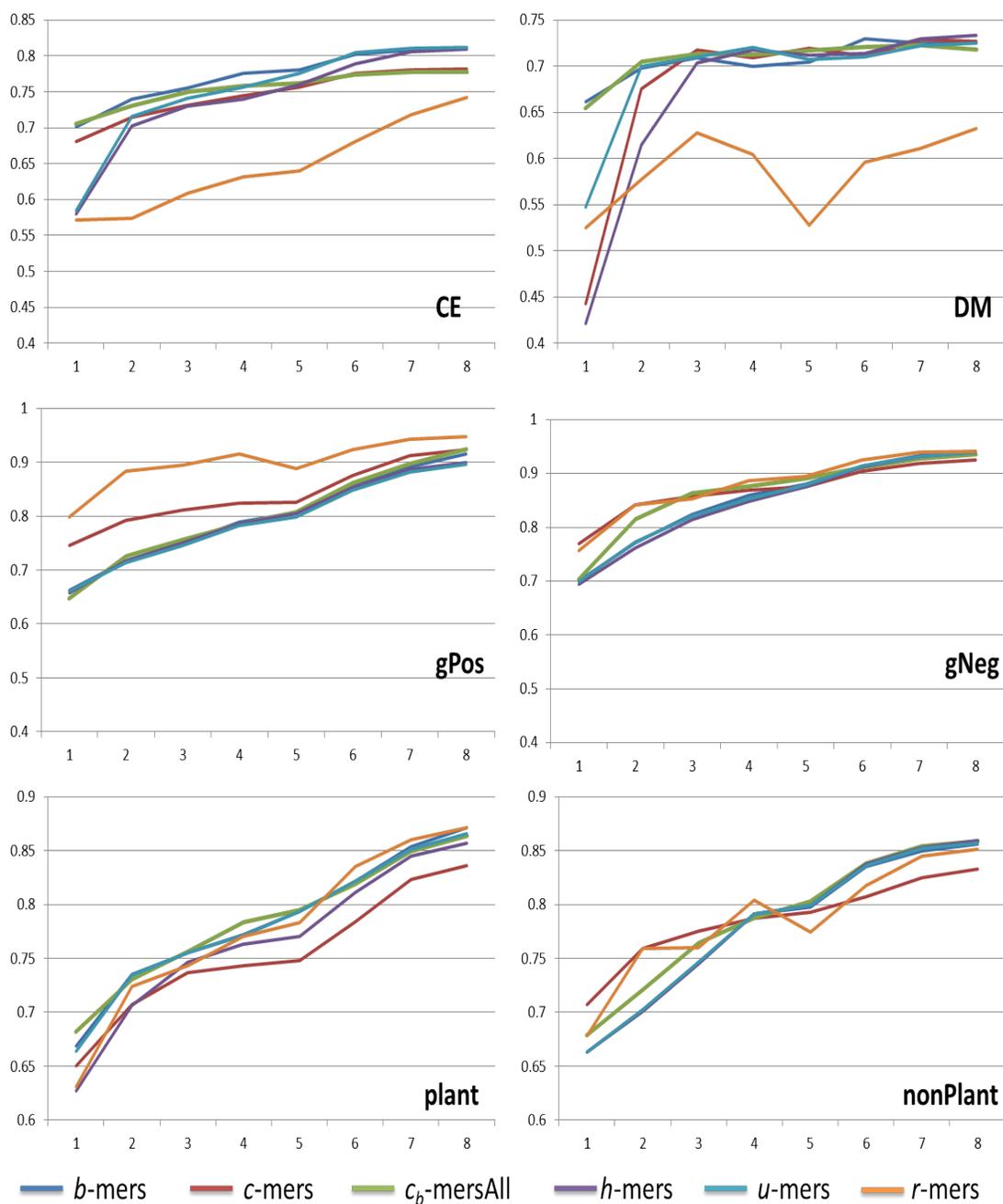


Figure 8.1: Variation of the performance of NBM classifier with the number of features selected from b-mers, c-mers, c_b-mers, h-mers and u-mers along with r-mers in supervised learning scenario is shown in the following graphs. In each graph, x-axis reports the number of features selected from each feature set, f , which is varied from 50 to 1500. y-axis reports the AUC values of NBM classifier learned from the respective feature sets. See Table 8.2 for the numbers shown on x-axis and the corresponding number of features selected, f .

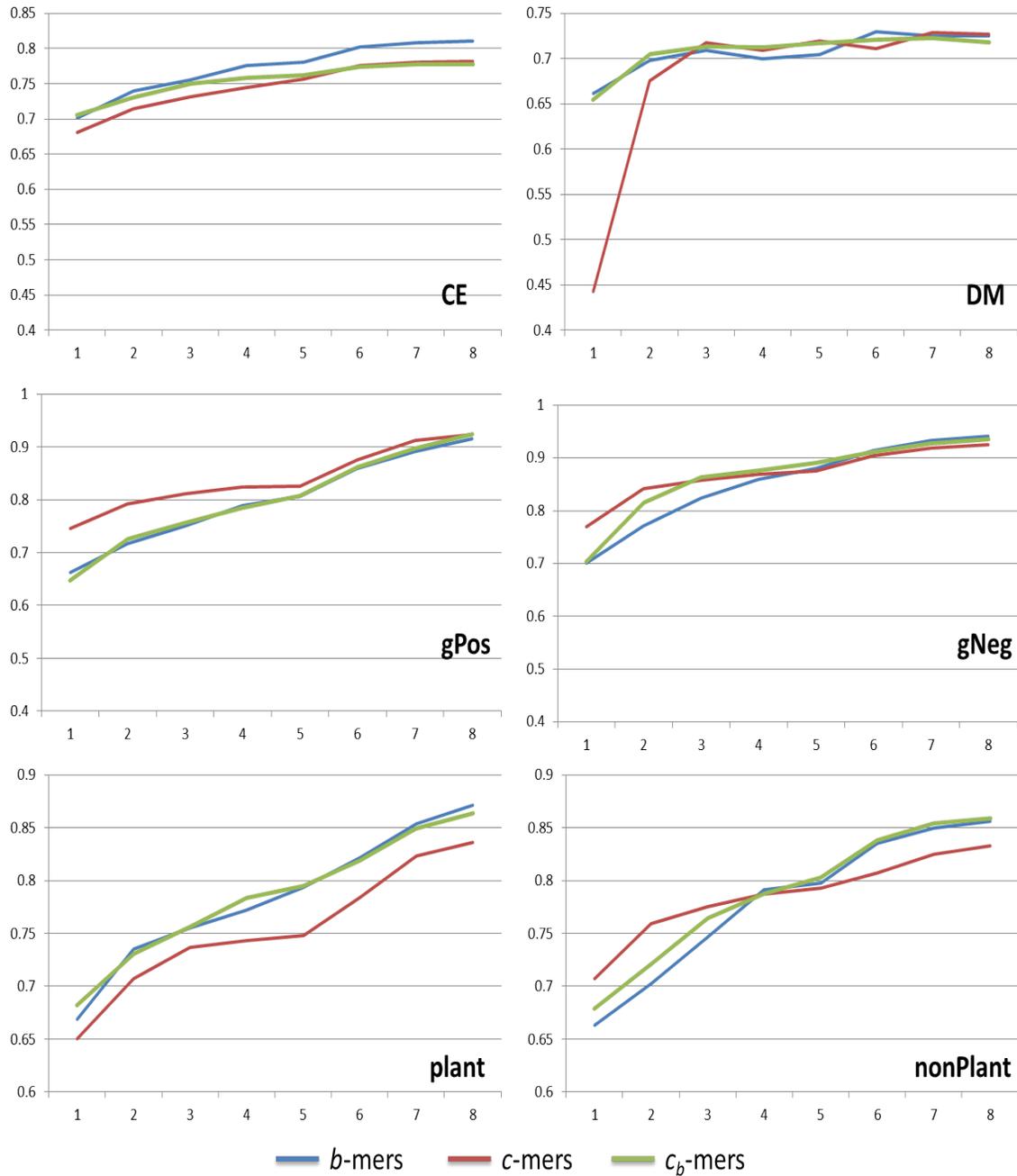


Figure 8.2: Variation of the performance of NBM classifier with the number of features selected from b -mers, c -mers and c_b -mers in supervised learning scenario is shown in the following graphs. In each graph, x -axis reports the number of features selected from each feature set, f , which is varied from 50 to 1500. y -axis reports the AUC values of NBM classifier learned from the respective feature sets. See Table 8.2 for the numbers shown on x -axis and the corresponding number of features selected, f .

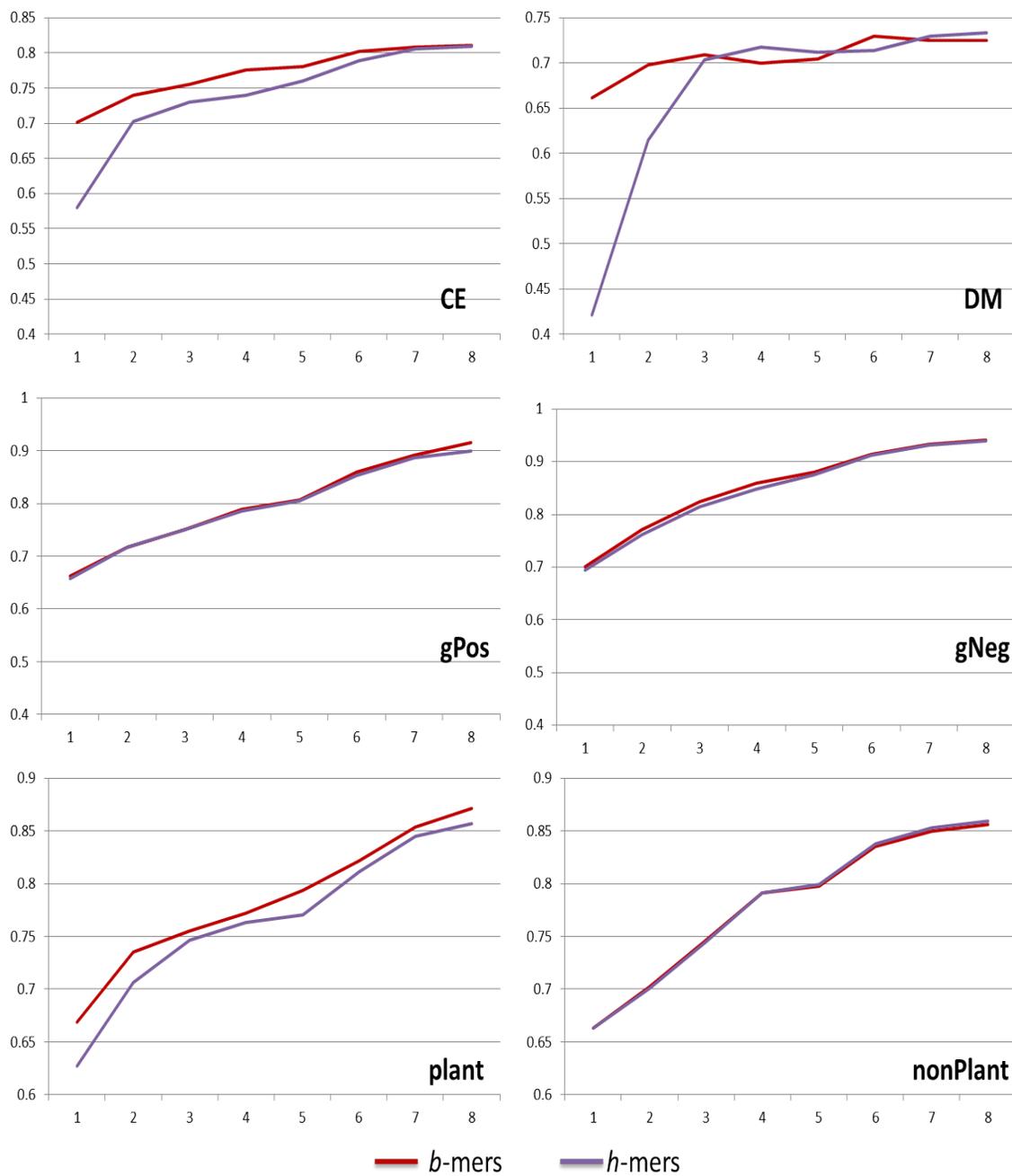


Figure 8.3: Variation of the performance of NBM classifier with the number of features selected from b-mers and h-mers in supervised learning scenario is shown in the following graphs. In each graph, x-axis reports the number of features selected from each feature set, f , which is varied from 50 to 1500. y-axis reports the AUC values of NBM classifier learned from the respective feature sets. See Table 8.2 for the numbers shown on x-axis and the corresponding number of features selected, f .

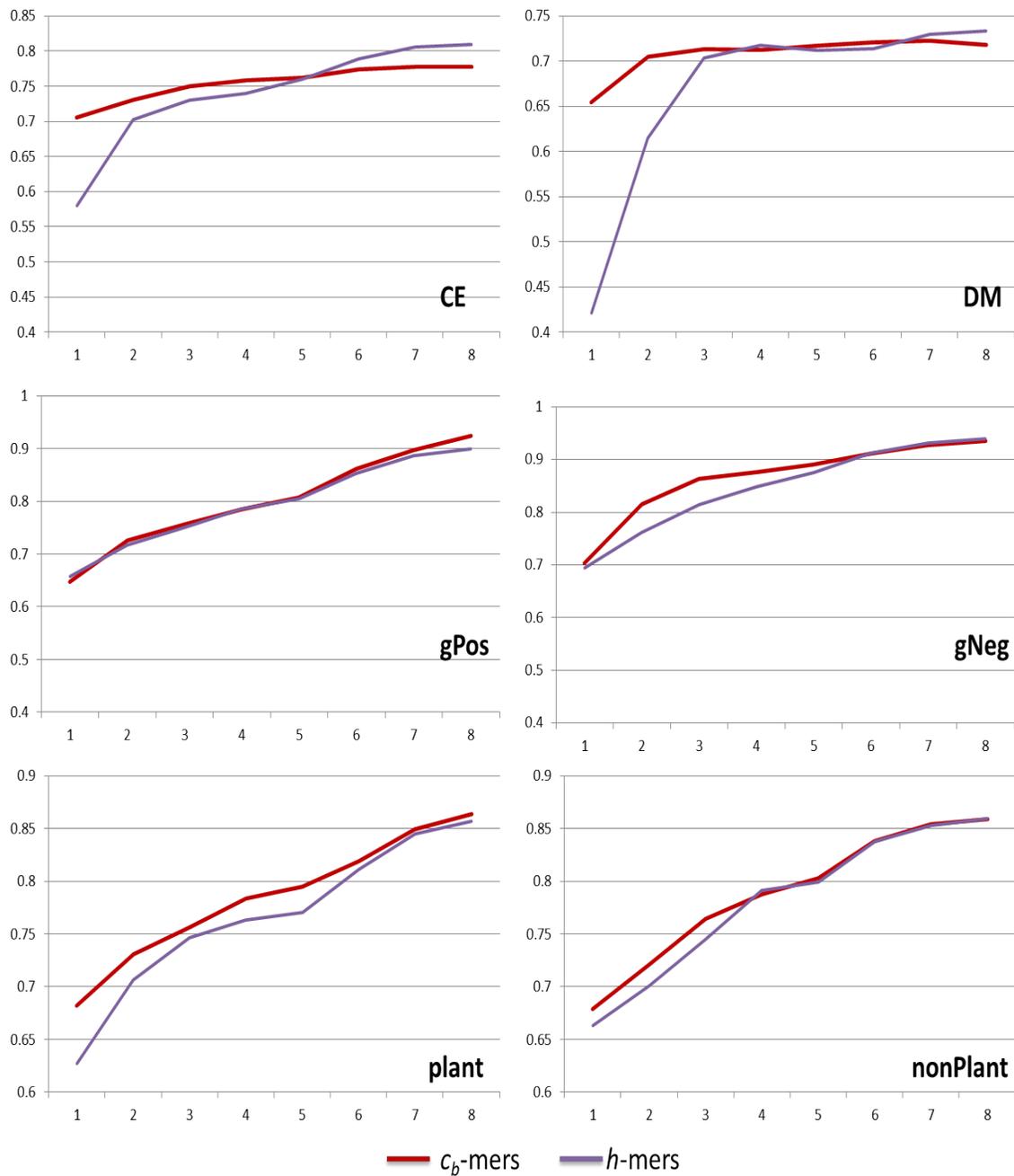


Figure 8.4: Variation of the performance of NBM classifier with the number of features selected from c_b -mers and h -mers, in supervised learning scenario is shown in the following graphs. In each graph, x-axis reports the number of features selected from each feature set, f , which is varied from 50 to 1500. y-axis reports the AUC values of NBM classifier learned from the respective feature sets. See Table 8.2 for the numbers shown on x-axis and the corresponding number of features selected, f .

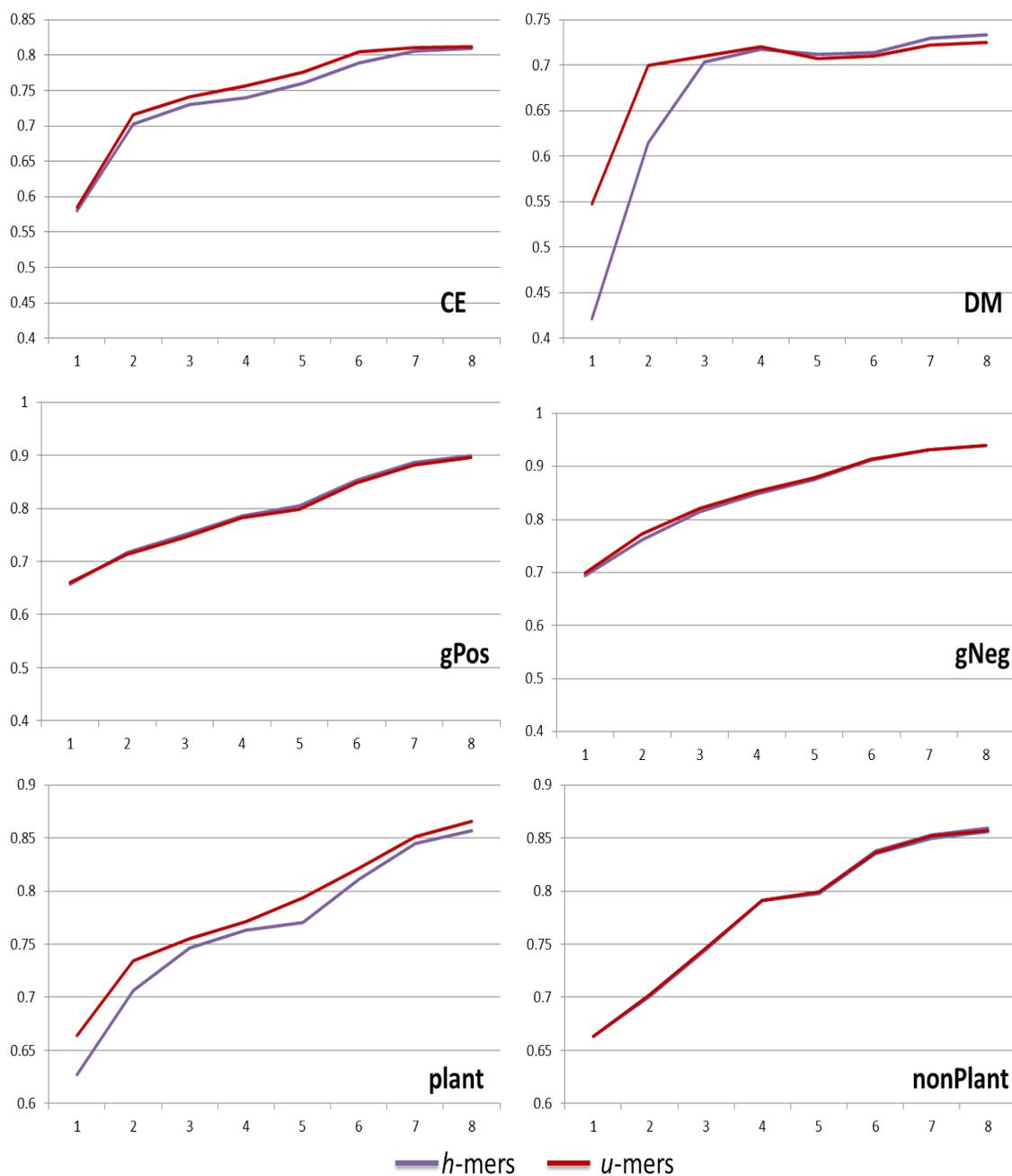


Figure 8.5: Variation of the performance of NBM classifier with the number of features selected from h-mers and u-mers, in supervised learning scenario is shown in the following graphs. In each graph, x-axis reports the number of features selected from each feature set, f , which is varied from 50 to 1500. y-axis reports the AUC values of NBM classifier learned from the respective feature sets. See Table 8.2 for the numbers shown on x-axis and the corresponding number of features selected, f .

Table 8.2: Variation of the performance of NBM classifier with the number of features selected from b -mers, c -mers, c_b -mers, h -mers and u -mers along with r -mers, with different number of features (selected using feature selection). For each dataset, maximum AUC among b -mers, c -mers, c_b -mers, u -mers, h -mers and r -mers for each variation of the amount of unlabeled data is reported in **bold font**.

Index	1	2	3	4	5	6	7	8
Number of features	50	100	150	200	250	500	1000	1500
CE(b -mers)	0.701	0.74	0.755	0.776	0.78	0.802	0.808	0.81
CE(c -mers)	0.681	0.715	0.731	0.745	0.757	0.776	0.781	0.782
CE(c_b -mers)	0.705	0.731	0.75	0.758	0.762	0.774	0.777	0.778
CE(u -mers)	0.58	0.702	0.73	0.74	0.76	0.789	0.806	0.809
CE(h -mers)	0.585	0.716	0.741	0.756	0.776	0.804	0.811	0.812
CE(r -mers)	0.571	0.574	0.609	0.632	0.64	0.681	0.718	0.742
DM(b -mers)	0.661	0.698	0.709	0.7	0.704	0.73	0.725	0.725
DM(c -mers)	0.443	0.675	0.717	0.709	0.719	0.711	0.729	0.727
DM(c_b -mers)	0.654	0.705	0.713	0.712	0.717	0.721	0.723	0.718
DM(u -mers)	0.421	0.615	0.703	0.717	0.712	0.714	0.73	0.733
DM(h -mers)	0.547	0.7	0.71	0.72	0.707	0.71	0.722	0.725
DM(r -mers)	0.525	0.577	0.628	0.604	0.528	0.596	0.611	0.632
gPos(b -mers)	0.662	0.716	0.75	0.789	0.807	0.859	0.891	0.915
gPos(c -mers)	0.746	0.792	0.812	0.824	0.826	0.876	0.912	0.923
gPos(c_b -mers)	0.647	0.725	0.756	0.785	0.808	0.861	0.897	0.924
gPos(u -mers)	0.658	0.717	0.751	0.786	0.805	0.853	0.887	0.9
gPos(h -mers)	0.66	0.713	0.745	0.783	0.798	0.848	0.881	0.896
gPos(r -mers)	0.798	0.884	0.894	0.915	0.888	0.924	0.942	0.948
gNeg(b -mers)	0.7	0.772	0.824	0.859	0.88	0.914	0.933	0.941
gNeg(c -mers)	0.77	0.841	0.857	0.869	0.876	0.904	0.918	0.925
gNeg(c_b -mers)	0.703	0.815	0.864	0.876	0.891	0.911	0.927	0.936
gNeg(u -mers)	0.695	0.761	0.815	0.848	0.875	0.912	0.931	0.939
gNeg(h -mers)	0.699	0.773	0.821	0.853	0.878	0.913	0.932	0.94
gNeg(r -mers)	0.757	0.842	0.853	0.887	0.895	0.925	0.94	0.941
plant(b -mers)	0.669	0.735	0.755	0.772	0.794	0.822	0.854	0.871
plant(c -mers)	0.65	0.707	0.737	0.743	0.748	0.784	0.823	0.836
plant(c_b -mers)	0.682	0.731	0.756	0.784	0.795	0.819	0.849	0.864
plant(u -mers)	0.627	0.706	0.746	0.763	0.77	0.811	0.845	0.857
plant(h -mers)	0.664	0.734	0.755	0.771	0.794	0.822	0.851	0.866
plant(r -mers)	0.631	0.724	0.743	0.77	0.783	0.835	0.86	0.871
nonPlant(b -mers)	0.663	0.702	0.746	0.791	0.798	0.835	0.85	0.856
nonPlant(c -mers)	0.707	0.759	0.775	0.787	0.793	0.807	0.825	0.833
nonPlant(c_b -mers)	0.679	0.721	0.764	0.788	0.803	0.838	0.854	0.859
nonPlant(u -mers)	0.663	0.701	0.745	0.791	0.799	0.838	0.853	0.859
nonPlant(h -mers)	0.663	0.702	0.746	0.791	0.799	0.836	0.852	0.857
nonPlant(r -mers)	0.678	0.759	0.76	0.804	0.774	0.818	0.845	0.851

8.3.3 Semi-supervised Learning

In order to analyze the predictive power of feature sets in semi-supervised learning scenario, Tables 8.3 and 8.4 present AUC values of ST and CT classifiers on all six datasets (CE, DM, gPos, gNeg, plant, and nonPlant), when the amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%. To better understand the behavior, results are also plotted in Figures 8.6 and 8.7. Although the general behavior and comparison among the feature sets remained the same as in the supervised learning scenario, the following observations were made specifically in semi-supervised learning scenario:

- **Nucleotide sequences:**

- In most cases, h -mers outperformed other feature sets with ST and CT classifiers. Furthermore, as opposed to supervised learning scenario, h -mers proved to be better than b -mers in semi-supervised learning scenario in most cases. One possible reason for this could be dimensionality of the features. h -mers, comprise of the entire set of b -mers and other similar subsequences. Therefore, the dimensionality of h -mers was greater than the dimensionality of b -mers. In semi-supervised learning scenario, the entire set of derived features (b -mers and h -mers, respectively) was used in order to represent the data, as opposed to selecting same number of features as in supervised learning experiments. If additional features (features present in h -mers but not in b -mers) captured using the CDA-based approach (on top of the BWT-based approach) were informative, and did not mislead the classifier, the performance of h -mers should be better than that of b -mers. The results confirmed this behavior, suggesting that the HBA approach successfully identified informative subsequences similar to b -mers, especially for nucleotide sequences.
- **r -mers:** Similar the performance in supervised learning scenario, r -mers in semi-supervised learning scenario rarely outperformed features generated using the

proposed approaches with nucleotide sequences.

- **ST versus CT:** By analyzing results in Tables 8.3 and 8.4, the conclusion was made that CT resulted in better AUC values compared to ST, in a majority of cases.

• **Protein sequences:** The results were less consistent with protein sequence datasets:

- For various datasets and classifiers, different feature sets were informative.
- In 41 out of 64 cases, with ST and CT combined, features generated using individual approaches (BWT and CDA-based approaches) outperformed the hybrid features (h -mers and u -mers).
- As opposed to behavior in the supervised learning scenario, the proposed approaches outperformed feature hashing in a majority of cases corresponding to all protein datasets. One reason for this behavior could be the lack of sufficient data to capture knowledge hidden within each bin. In hashing, k -mers are grouped together, and without sufficient amounts of labeled data, the classifier may not accurately estimate the relation between feature values and classes.

8.3.4 Domain Adaptation Scenario

In order to analyze the predictive power of feature sets in the domain adaptation learning scenario, Table 8.5 presents AUC values of the NBM for domain adaptation classifier on all six combinations of source→target data (CE→DM, DM→CE, gNeg→gPos, gPos→gNeg, nonPlant→plant, and plant→nonPlant) when the amount of target labeled data is fixed to 20% and the amount of target unlabeled data was varied from 20% to 80%. To better understand the behavior, results are also plotted in Figures 8.8. Although h -mers rarely outperformed b -mers and c -mers, features generated using individual approaches (b -mers, c -mers, and c_b -mers) outperformed hybrid features (h -mers and u -mers) in majority of the cases (22 out of 24).

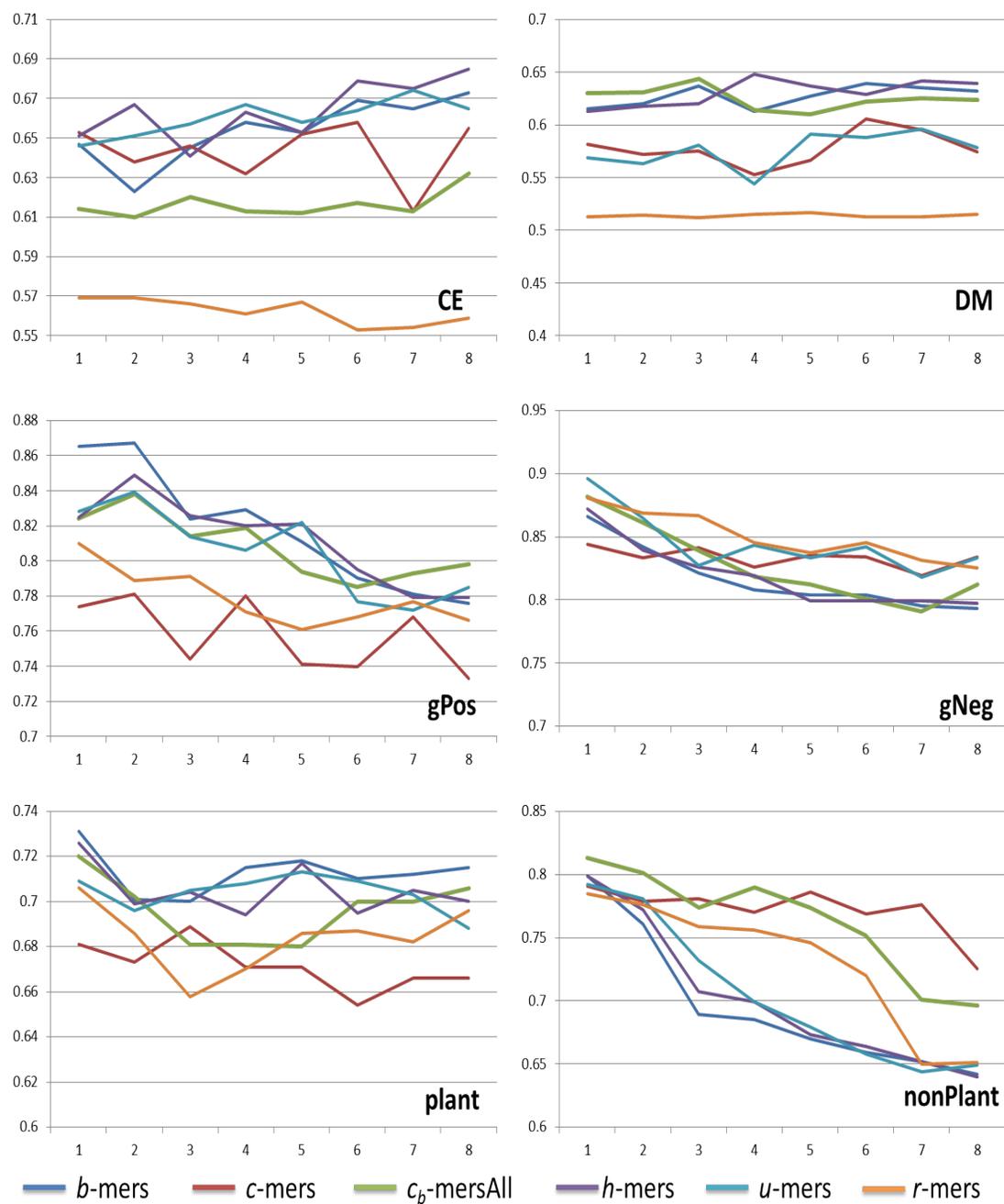


Figure 8.6: Variation of the performance with the amount of unlabeled data used with *ST* classifier, learned using *b*-mers, *c*-mers, *c_b*-mers, *h*-mers, *u*-mers and *r*-mers. Each graph plots the AUC values (on *y*-axis) for *CE*, *DM*, *gPos*, *gNeg*, *plant*, and *nonPlant* datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on *x*-axis). See Table 8.3 for the numbers shown on *x*-axis and the corresponding percentage of unlabeled data used to conduct experiments.

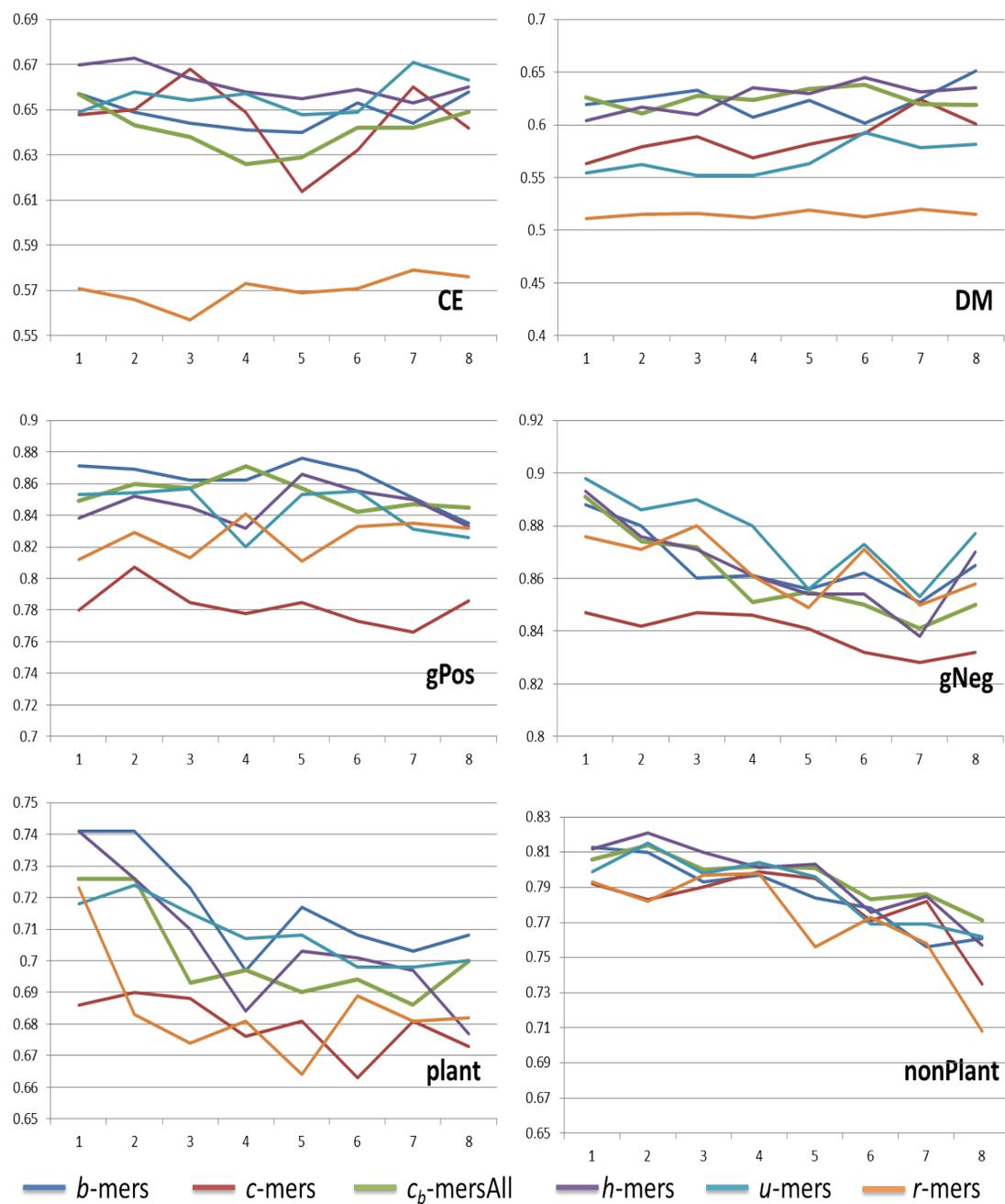


Figure 8.7: Variation of the performance with the amount of unlabeled data used with CT classifier, learned using b-mers, c-mers, c_b-mers, h-mers, u-mers and r-mers. Each graph plots the AUC values (on y-axis) for CE, DM, gPos, gNeg, plant, and nonPlant datasets when the amount of labeled data was fixed to 10%, while the amount of unlabeled data was varied from 20% to 90% (on x-axis). See Table 8.4 for the numbers shown on x-axis and the corresponding percentage of unlabeled data used to conduct experiments.

Table 8.3: AUC values obtained with ST classifier learned using b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers. The amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%. For each dataset, maximum AUC among b -mers, c -mers, c_b -mers, u -mers, h -mers and r -mers for each variation of the amount of unlabeled data is reported in **bold font**.

Index	1	2	3	4	5	6	7	8
Unlab	20%	30%	40%	50%	60%	70%	80%	90%
CE(b -mers)	0.647	0.623	0.645	0.658	0.653	0.669	0.665	0.673
CE(c -mers)	0.653	0.638	0.646	0.632	0.652	0.658	0.613	0.655
CE(c_b -mers)	0.614	0.61	0.62	0.613	0.612	0.617	0.613	0.632
CE(u -mers)	0.646	0.651	0.657	0.667	0.658	0.664	0.674	0.665
CE(h -mers)	0.651	0.667	0.641	0.663	0.653	0.679	0.675	0.685
CE(r -mers)	0.569	0.569	0.566	0.561	0.567	0.553	0.554	0.559
DM(b -mers)	0.615	0.62	0.637	0.613	0.627	0.639	0.635	0.632
DM(c -mers)	0.582	0.572	0.575	0.553	0.566	0.606	0.595	0.574
DM(c_b -mers)	0.63	0.631	0.644	0.614	0.61	0.622	0.625	0.624
DM(u -mers)	0.569	0.563	0.581	0.544	0.591	0.588	0.596	0.578
DM(h -mers)	0.613	0.618	0.62	0.648	0.637	0.629	0.642	0.639
DM(r -mers)	0.513	0.514	0.512	0.515	0.517	0.513	0.513	0.515
gPos(b -mers)	0.865	0.867	0.824	0.829	0.811	0.79	0.781	0.776
gPos(c -mers)	0.774	0.781	0.744	0.78	0.741	0.74	0.768	0.733
gPos(c_b -mers)	0.824	0.838	0.814	0.819	0.794	0.785	0.793	0.798
gPos(u -mers)	0.828	0.839	0.814	0.806	0.822	0.777	0.772	0.785
gPos(h -mers)	0.825	0.849	0.826	0.82	0.821	0.795	0.779	0.779
gPos(r -mers)	0.81	0.789	0.791	0.771	0.761	0.768	0.777	0.766
gNeg(b -mers)	0.866	0.842	0.821	0.808	0.804	0.804	0.795	0.793
gNeg(c -mers)	0.844	0.833	0.841	0.826	0.835	0.834	0.819	0.834
gNeg(c_b -mers)	0.882	0.861	0.839	0.818	0.812	0.801	0.791	0.812
gNeg(u -mers)	0.896	0.865	0.827	0.843	0.833	0.842	0.818	0.833
gNeg(h -mers)	0.872	0.839	0.826	0.819	0.799	0.799	0.799	0.797
gNeg(r -mers)	0.881	0.869	0.867	0.845	0.837	0.845	0.831	0.825
plant(b -mers)	0.731	0.701	0.7	0.715	0.718	0.71	0.712	0.715
plant(c -mers)	0.681	0.673	0.689	0.671	0.671	0.654	0.666	0.666
plant(c_b -mers)	0.72	0.702	0.681	0.681	0.68	0.7	0.7	0.706
plant(u -mers)	0.709	0.696	0.705	0.708	0.713	0.709	0.703	0.688
plant(h -mers)	0.726	0.699	0.704	0.694	0.717	0.695	0.705	0.7
plant(r -mers)	0.706	0.686	0.658	0.67	0.686	0.687	0.682	0.696
nonPlant(b -mers)	0.799	0.761	0.689	0.685	0.67	0.659	0.652	0.642
nonPlant(c -mers)	0.791	0.779	0.781	0.77	0.786	0.769	0.776	0.725
nonPlant(c_b -mers)	0.813	0.801	0.774	0.79	0.774	0.752	0.701	0.696
nonPlant(u -mers)	0.792	0.781	0.732	0.699	0.679	0.658	0.644	0.649
nonPlant(h -mers)	0.799	0.772	0.707	0.699	0.673	0.664	0.652	0.64
nonPlant(r -mers)	0.785	0.776	0.759	0.756	0.746	0.72	0.65	0.651

Table 8.4: AUC values obtained with CT classifier learned using b -mers, c -mers, c_b -mers, h -mers, u -mers and r -mers. The amount of labeled data was fixed to 10% and the amount of unlabeled data was varied from 20% to 90%. For each dataset, maximum AUC among b -mers, c -mers, c_b -mers, u -mers, h -mers and r -mers for each variation of the amount of unlabeled data is reported in **bold font**.

Index	1	2	3	4	5	6	7	8
Unlab	20%	30%	40%	50%	60%	70%	80%	90%
CE(b -mers)	0.657	0.649	0.644	0.641	0.64	0.653	0.644	0.658
CE(c -mers)	0.648	0.65	0.668	0.649	0.614	0.632	0.66	0.642
CE(c_b -mers)	0.657	0.643	0.638	0.626	0.629	0.642	0.642	0.649
CE(u -mers)	0.649	0.658	0.654	0.657	0.648	0.649	0.671	0.663
CE(h -mers)	0.67	0.673	0.664	0.658	0.655	0.659	0.653	0.66
CE(r -mers)	0.571	0.566	0.557	0.573	0.569	0.571	0.579	0.576
DM(b -mers)	0.619	0.626	0.633	0.607	0.623	0.602	0.625	0.651
DM(c -mers)	0.563	0.579	0.589	0.569	0.582	0.592	0.624	0.601
DM(c_b -mers)	0.626	0.611	0.628	0.624	0.634	0.638	0.62	0.619
DM(u -mers)	0.554	0.562	0.552	0.552	0.563	0.593	0.578	0.582
DM(h -mers)	0.604	0.617	0.61	0.635	0.63	0.645	0.631	0.635
DM(r -mers)	0.511	0.515	0.516	0.512	0.519	0.513	0.52	0.515
gPos(b -mers)	0.871	0.869	0.862	0.862	0.876	0.868	0.851	0.835
gPos(c -mers)	0.78	0.807	0.785	0.778	0.785	0.773	0.766	0.786
gPos(c_b -mers)	0.849	0.86	0.857	0.871	0.857	0.842	0.847	0.845
gPos(u -mers)	0.853	0.854	0.857	0.82	0.853	0.855	0.831	0.826
gPos(h -mers)	0.838	0.852	0.845	0.832	0.866	0.855	0.85	0.833
gPos(r -mers)	0.812	0.829	0.813	0.841	0.811	0.833	0.835	0.832
gNeg(b -mers)	0.888	0.88	0.86	0.861	0.856	0.862	0.851	0.865
gNeg(c -mers)	0.847	0.842	0.847	0.846	0.841	0.832	0.828	0.832
gNeg(c_b -mers)	0.891	0.874	0.872	0.851	0.855	0.85	0.841	0.85
gNeg(u -mers)	0.898	0.886	0.89	0.88	0.856	0.873	0.853	0.877
gNeg(h -mers)	0.893	0.876	0.871	0.861	0.854	0.854	0.838	0.87
gNeg(r -mers)	0.876	0.871	0.88	0.861	0.849	0.871	0.85	0.858
plant(b -mers)	0.741	0.741	0.723	0.697	0.717	0.708	0.703	0.708
plant(c -mers)	0.686	0.69	0.688	0.676	0.681	0.663	0.681	0.673
plant(c_b -mers)	0.726	0.726	0.693	0.697	0.69	0.694	0.686	0.7
plant(u -mers)	0.718	0.724	0.715	0.707	0.708	0.698	0.698	0.7
plant(h -mers)	0.741	0.726	0.71	0.684	0.703	0.701	0.697	0.677
plant(r -mers)	0.723	0.683	0.674	0.681	0.664	0.689	0.681	0.682
nonPlant(b -mers)	0.813	0.81	0.793	0.797	0.784	0.778	0.756	0.761
nonPlant(c -mers)	0.792	0.783	0.79	0.799	0.795	0.771	0.782	0.735
nonPlant(c_b -mers)	0.806	0.814	0.8	0.802	0.801	0.783	0.786	0.771
nonPlant(u -mers)	0.799	0.815	0.798	0.804	0.796	0.769	0.769	0.762
nonPlant(h -mers)	0.812	0.821	0.81	0.801	0.803	0.776	0.785	0.757
nonPlant(r -mers)	0.793	0.782	0.797	0.798	0.756	0.773	0.758	0.708

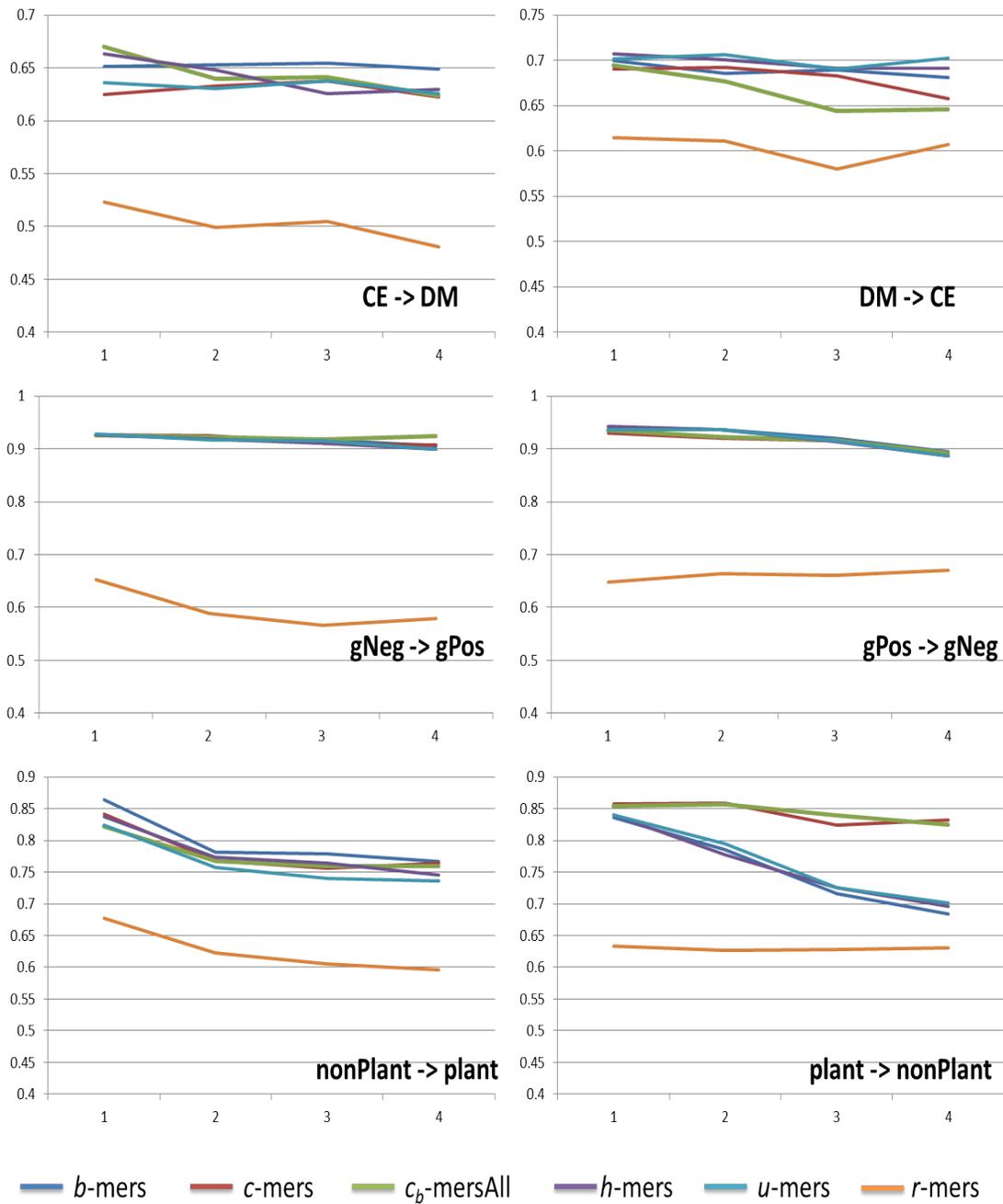


Figure 8.8: Variation of the performance with the amount of unlabeled data used with NBM for domain adaptation classifier, learned using *b*-mers, *c*-mers, *c_b*-mers, *h*-mers, *u*-mers and *r*-mers, respectively. Each graph plots the AUC values (on *y*-axis) for *CE*→*DM*, *DM*→*CE*, *gPos*→*gNeg*, *gNeg*→*gPos*, *plant*→*nonPlant*, and *nonPlant*→*plant* combinations of source→target datasets when the amount of labeled data was fixed to 20%, while the amount of unlabeled data was varied from 20% to 80% (on *x*-axis). See Table 8.5 for the numbers shown on *x*-axis and the corresponding percentage of unlabeled data used to conduct experiments.

Table 8.5: Results for the variation of unlabeled data with NBM for domain adaptation algorithm when learned using various feature sets: *b*-mers, *c*-mers, *c_b*-mers, *u*-mers, *h*-mers, *r*-mers and *k*-mers. Table includes AUC values for the following combinations of source and target datasets: *CE*→*DM*, *DM*→*CE*, *gPos*→*gNeg*, *gNeg*→*gPos*, *plant*→*nonPlant*, and *nonPlant*→*plant*. For all experiments, the amount of labeled data is fixed to 20%. For each dataset, maximum AUC among *b*-mers, *c*-mers, *c_b*-mers, *u*-mers, *h*-mers and *r*-mers for each variation of the amount of unlabeled data is reported in **bold font**.

Source→Target	Unlabeled Data	20%	40%	60%	80%
CE→DM	<i>b</i> -mers	0.6514	0.6528	0.6548	0.649
	<i>c</i> -mers	0.6248	0.6332	0.6378	0.6222
	<i>c_b</i> -mers	0.6698	0.6394	0.6412	0.6234
	<i>u</i> -mers	0.636	0.6304	0.638	0.6256
	<i>h</i> -mers	0.6634	0.648	0.6256	0.63
	<i>r</i> -mers	0.5228	0.499	0.5044	0.4804
DM→CE	<i>b</i> -mers	0.6996	0.6852	0.6892	0.6812
	<i>c</i> -mers	0.69	0.692	0.6828	0.658
	<i>c_b</i> -mers	0.6948	0.6768	0.644	0.6462
	<i>u</i> -mers	0.702	0.7066	0.6906	0.7028
	<i>h</i> -mers	0.7072	0.701	0.6912	0.6916
	<i>r</i> -mers	0.6144	0.611	0.5802	0.6072
gNeg→gPos	<i>b</i> -mers	0.9268	0.9202	0.917	0.9054
	<i>c</i> -mers	0.9258	0.9246	0.91	0.9074
	<i>c_b</i> -mers	0.9252	0.922	0.9184	0.9234
	<i>u</i> -mers	0.928	0.917	0.915	0.8988
	<i>h</i> -mers	0.9266	0.9178	0.9098	0.8992
	<i>r</i> -mers	0.6534	0.5886	0.5658	0.5786
gPos→gNeg	<i>b</i> -mers	0.9414	0.936	0.9202	0.8938
	<i>c</i> -mers	0.9294	0.9206	0.9154	0.8884
	<i>c_b</i> -mers	0.9352	0.9232	0.9162	0.8918
	<i>u</i> -mers	0.9362	0.9354	0.915	0.8858
	<i>h</i> -mers	0.9424	0.9356	0.9132	0.886
	<i>r</i> -mers	0.6476	0.664	0.6612	0.6696
nonPlant→plant	<i>b</i> -mers	0.8642	0.7808	0.7788	0.7662
	<i>c</i> -mers	0.8412	0.7684	0.756	0.7636
	<i>c_b</i> -mers	0.8224	0.7672	0.7588	0.7594
	<i>u</i> -mers	0.824	0.758	0.74	0.7364
	<i>h</i> -mers	0.838	0.7736	0.7636	0.746
	<i>r</i> -mers	0.6778	0.6226	0.6054	0.5958
plant→nonPlant	<i>b</i> -mers	0.8358	0.7856	0.7154	0.6844
	<i>c</i> -mers	0.8574	0.8586	0.8242	0.8316
	<i>c_b</i> -mers	0.854	0.8568	0.839	0.8246
	<i>u</i> -mers	0.84	0.7954	0.7256	0.7012
	<i>h</i> -mers	0.8398	0.7774	0.7258	0.6964
	<i>r</i> -mers	0.6334	0.6268	0.6282	0.631

Chapter 9

Conclusion and Future Work

Section 9.1 of this chapter draws conclusions for the work presented in this dissertation and discuss limitations of the approaches proposed to construct sequential features. Improvements and future directions for this work are proposed in Section 9.2.

9.1 Conclusion

This work proposed three novel unsupervised approaches to construct sequential features, specifically for biological sequence classification problems. Section 9.1.1 outlines the contributions of this dissertation. The merits of the proposed approaches are presented in Section 9.1.2 and limitations are detailed in Section 9.1.3.

9.1.1 Contributions

- Burrows Wheeler Transform-based approach, which takes into account the length and suffix information to construct sequential features (b -mers) that occur multiple times in at least one sequence [Tangirala and Caragea, 2014b,c; Herndon et al., 2014].
- Community detection-based approach, that uses community detection algorithm to identify communities. In this approach, communities are defined as a group of sim-

ilar subsequences of a particular length, further refined to form motifs. The unique subsequences of the top motifs (c -mers) are used as features [Tangirala and Caragea, 2014a; Tangirala et al., 2015].

- Hybrid approach, that combines the BWT-based approach with the CDA-based approach to generate b -mers with certain mismatches that span various sequences. A paper on HBA approach and its comparison with BWT and CDA-based approaches is currently under preparation.

Experiments were conducted to evaluate parameters corresponding to each of the proposed approach and to evaluate the predictive power of features generated using the proposed approaches. Experiments were conducted in three learning scenarios: supervised, semi-supervised, and domain adaptation learning scenarios.

9.1.2 Merits

- **Burrows Wheeler Transform-based approach:** This approach can be considered as a dimensionality reduction technique because the features obtained through BWT represent a subset of the set of k -mers, generated using a sliding window-based approach. To the best of the author’s knowledge, BWT has never been used to generate features for biological sequence classification problems. The approach is scalable in terms of the number of sequences because it constructs features from each sequence, as opposed to a group of sequences. Results of experiments on nucleotide and protein datasets showed that the use of BWT to generate features reduced the size of input feature space while retaining many informative features in all three learning scenarios. Furthermore, BWT-based approach does not have many notable/significant parameters to be tuned.
- **Community detection-based approach:** Similar to the BWT-based approach, this approach also constructs a reduced set of k -mers, generated using a sliding window-

based approach. Features constructed using the CDA-based approach satisfy the ZOMOMPS constraint. Because the running time of the community detection approach is highly dependent on the total number of sequences and length of the sequences, a model that uses an existing community detection based approach on a set of randomly selected samples of sequences was proposed. Experimental results show that the proposed approach, when invoked using small samples (less sequences - small running time) produced better results compared to invoking it on larger samples (more sequences - large running time). Furthermore, as opposed to the original approach of using Hamming distance to construct the network and identify the motifs, a novel idea of using substitution scores as a similarity metric to identify motifs for protein sequences was proposed. Results of the experiments in three learning scenarios showed that the proposed approach generated low-dimensional informative features in supervised, semi-supervised, and domain adaptation scenarios.

- **Hybrid approach:** This approach was introduced in an attempt to improve the performance of b -mers, especially for nucleotide sequences, and to generate b -mers with certain mismatches that span various sequences. Experiments, similar to other approaches, were conducted in all three learning scenarios on protein and nucleotide sequence datasets. The results of the experiments showed that the HBA approach outperformed BWT and CDA-based approaches in semi-supervised learning scenario, for nucleotide sequences.

9.1.3 Limitations

The proposed approaches present the following limitations:

- Although the BWT-based approach is scalable in terms of the number of sequences, the running time to construct b -mers is largely affected by the length of the sequences. Sorting all possible permutations was performed for each sequence, requiring quadratic

time and space in the length of the sequence.

- All three proposed approaches are susceptible to skewed data. The BWT-based approach constructs features per sequence, and the other two approaches construct features from samples of sequences. When the data is skewed, each sample is also likely skewed. The use of most sequences (for BWT-based) or samples that contain a majority of sequences (for the CDA-based and HBA approaches) corresponding to one specific class may result in features that are informative only to that class.

9.2 Future Work

- BWT has been successfully used with several tree data structures to efficiently identify repeats from biological sequences. Exploration should be made to determine if such data structures can be used to improve the run time of the BWT-based approach, especially for very long sequences.
- Different community detection algorithms may discover different community structures from the same network. Investigation should be made into understanding the effects of various community detection algorithms in generating sequential features for biological sequence classification problems using the CDA-based approach.
- Because proposed approaches are highly susceptible to skewed data, the addition of a wrapper on top of the proposed approaches, that constructs a set of balanced data samples and further construct features from those balanced samples, as opposed to using all the sequences would be interesting.
- Furthermore, investigation should be made into evaluating the predictive power of the proposed feature sets when generated using sequences corresponding to the positive class alone. However, the resulting approach will not be unsupervised in nature.

- Furthermore, *b*-mers, *c*-mers, and *h*-mers should be compared to motifs generated using existing tools such as MEME [Bailey et al., 2009].
- Also, investigation should be made into extending the proposed features to other datasets and sequence classification problems such as identifying protein functions or structures, identifying splice sites, etc.,.
- It would be interesting to extend proposed approaches to identify motifs related to other biological problems. For example, identifying transcription factor binding sites in a set of nucleotide sequences.

Bibliography

- Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering, ICDE '95*, pages 3–14, Washington, DC, USA. IEEE Computer Society.
- Allan, E. and Wren, B. (2003). Genes to genetic immunization: identification of bacterial vaccine candidates. *Methods*, 31(3):193–198.
- Bailey, T. and Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80.
- Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., Ren, J., Li, W. W., and Noble, W. S. (2009). MEME Suite: tools for motif discovery and searching. *Nucleic Acids Research*, 37(suppl 2):W202–W208.
- Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5:537–550.
- Becher, V., Deymonnaz, A., and Heiber, P. (2009). Efficient computation of all perfect repeats in genomic sequences of up to half a gigabyte, with a case study on the human genome. *Bioinformatics*, 25(14):1746–1753.
- Bellier, B., Six, A., Thomas-Vaslin, V., and Klatzmann, D. (2013). Reverse vaccinology. In Dubitzky, W., Wolkenhauer, O., Cho, K.-H., and Yokota, H., editors, *Encyclopedia of Systems Biology*, pages 1856–1856. Springer New York.
- Black, D. L. (2003). Mechanisms of alternative pre-messenger RNA splicing. *Annual Review of Biochemistry*, 72:291–336.

- Blondel, V., Guillaume, J., Lambiotte, R., and Mech, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech*, page P10008.
- Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98*, pages 92–100, New York, NY, USA. ACM.
- Boucher, C. and King, J. (2010). Fast motif recognition via application of statistical thresholds. *BMC Bioinformatics*, 11(Suppl 1).
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- Bucher, P. (1990). Weight matrix descriptions of 4 eukaryotic RNA polymerase-ii promoter elements derived from 502 unrelated promoter sequences. *Journal of Molecular Biology*, 212:563–578.
- Buhler, J. and Tompa, M. (2001). Finding motifs using random projections. In *Proceedings of the Fifth Annual International Conference on Computational Biology, RECOMB '01*, pages 69–76, New York, NY, USA. ACM.
- Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm. Technical Report 124.
- Caragea, C., Silvescu, A., and Mitra, P. (2012). Protein sequence classification using feature hashing. *Proteome Science*, 10(1):1–8.
- Caropreso, M. F., Matwin, S., and Sebastiani, F. (2001). Text databases and document management. chapter A Learner-independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization, pages 78–102. IGI Global, Hershey, PA, USA.
- Cheeseman, P. and Stutz, J. (1996). Advances in knowledge discovery and data mining.

- chapter In *Bayesian Classification (AutoClass): Theory and Results*, pages 153–180. American Association for Artificial Intelligence, Menlo Park, CA, USA.
- Chen, L. and Liu, W. (2013). Frequent patterns mining in multiple biological sequences. *Computers in Biology and Medicine*, 43(10):1444 – 1452.
- Cheng, B. Y. M., Carbonell, J. G., and Klein-Seetharaman, J. (2005). Protein classification based on text document classification techniques. *Proteins: Structure, Function, and Bioinformatics*, 58(4):955–970.
- Chuzhanova, N. A., Jones, A. J., and Margetts, S. (1998). Feature selection for genetic sequence classification. *Bioinformatics*, 14(2):139–143.
- Clauset, A., Newman, M. E. J., , and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, pages 1– 6.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.
- Danek, A., Pokrzywa, R., Makaowska, I., and Polaski, A. (2012). Application of the Burrows-Wheeler transform for searching for approximate tandem repeats. In Shibuya, T., Kashima, H., Sese, J., and Ahmad, S., editors, *Pattern Recognition in Bioinformatics*, volume 7632 of *Lecture Notes in Computer Science*, pages 255–266. Springer Berlin Heidelberg.
- Das, M. and Dai, H.-K. (2007). A survey of DNA motif finding algorithms. *BMC Bioinformatics*, 8(Suppl 7).
- Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978). A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5(suppl 3):345–351.
- Degroeve, S., De Baets, B., Van de Peer, Y., and Rouzé, P. (2002). Feature subset selection for splice site prediction. *Bioinformatics*, 18(suppl 2):S75–S83.

- Donetti, L. and Muñoz, M. A. (2005). Improved spectral algorithm for the detection of network communities. In *Proceedings of the 8th Granada Seminar - Computational and Statistical Physics*, pages 1–2.
- Dongfang, N. and Xiaolong, Z. (2013). Prediction of hot regions in protein-protein interactions based on complex network and community detection. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 17–23.
- Down, T. A. and Hubbard, T. J. P. (2005). NestedMICA: sensitive inference of over-represented motifs in nucleic acid sequence. *Nucleic Acids Research*, 33(5):1445–1453.
- Emanuelsson, O., Nielsen, H., Brunak, S., and Heijne, G. (2000). Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of Molecular Biology*, 300(4):1005–1016.
- Eskin, E. and Pevzner, P. A. (2002). Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18(suppl 1):S354–S363.
- Exarchos, T., Papaloukas, C., Lampros, C., and Fotiadis, D. (2006). Protein classification using sequential pattern mining. In *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '06.*, pages 5814–5817.
- Exarchos, T. P., Tsipouras, M. G., Papaloukas, C., and Fotiadis, D. I. (2008). A two-stage methodology for sequence classification based on sequential pattern mining and optimization. *Data and Knowledge Engineering*, 66(3):467 – 487.
- Ferragina, P. and Manzini, G. (2000). Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 390–398.
- Forman, G. and Kirshenbaum, E. (2008). Extremely fast text feature extraction for classi-

- fication and indexing. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1221–1230, New York, NY, USA. ACM.
- Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3):75 – 174.
- Fortunato, S. and Lancichinetti, A. (2009). Community detection algorithms: A comparative analysis: Invited presentation, extended abstract. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 27:1–27:2, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- Fotiadis, D., Exarchos, T., Tsipouras, M., and Papaloukas, C. (2007). Biosequence classification using sequential pattern mining and optimization. In *6th International Special Topic Conference on Information Technology Applications in Biomedicine, 2007. ITAB 2007.*, pages 58–61.
- Galavotti, L., Sebastiani, F., and Simi, M. (2000). Experiments on the use of feature selection and negative evidence in automated text categorization. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '00, pages 59–68, London, UK, UK. Springer-Verlag.
- Gardy, J. L., Laird, M. R., Chen, F., Rey, S., Walsh, C. J., Ester, M., and Brinkman, F. S. L. (2005). PSORTb v.2.0: Expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis. *Bioinformatics*, 21(5):617–623.
- Gerstein, M. B., Bruce, C., Rozowsky, J. S., Zheng, D., Du, J., Korb, J. O., Emanuelsson, O., Zhang, Z. D., Weissman, S., and Snyder, M. (2007). What is a gene, post-ENCODE? History and updated definition. *Genome Research*, 17(6):669–681.
- Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

- Griffith, M., Tang, M. J., Griffith, O. L., Morin, R. D., Chan, S. Y., Asano, J. K., Zeng, T., Flibotte, S., Ally, A., Baross, A., Hirst, M., Jones, S. J. M., Morin, G. B., Tai, I. T., and Marra, M. A. (2008). ALEXA: a microarray design platform for alternative expression analysis. *Nature Methods*, 5(2):118.
- Guimera, R. and Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900.
- Guimera, R., S.-P. M. A. L. (2004). Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:art. no. 025101.
- Hanchuan, P., Fuhui, L., and Chris, D. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1226–1238.
- Harenberg, S., Bello, G., Gjeltema, L., Ranshous, S., Harlalka, J., Seay, R., Padmanabhan, K., and Samatova, N. (2014). Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):426–439.
- He, Y., Wu, X., Zhu, X., and Arslan, A. (2007). Mining frequent patterns with wildcards from biological sequences. In *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2007.*, pages 329–334.
- Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919.
- Herndon, N., Tangirala, K., and Caragea, D. (2014). Predicting Protein Localization Using a Domain Adaptation Naive Bayes Classifier with Burrows Wheeler Transform Features. In *Proceedings of the 6th IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2014*, pages 501–504.

- Huang, J., Lu, J., and Ling, L. C. X. (2003). Comparing Naïve Bayes, decision trees, and SVM with AUC and accuracy. In *Proceedings of the Third IEEE International Conference on Data Mining, ICDM 2003*, pages 553–556. IEEE Computer Society.
- Jason, D. M. R., Lawrence, S., Jaime, T., and David, R. K. (2003). Tackling the Poor Assumptions of Naïve Bayes Text Classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623.
- Jia, C., Carson, M., and Yu, J. (2013). A fast weak motif-finding algorithm based on community detection in graphs. *BMC Bioinformatics*, 14(1):1–14.
- Kang, T. H., soo Yoo, J., and Kim, H. Y. (2007). Mining frequent contiguous sequence patterns in biological sequences. In *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, 2007. BIBE 2007.*, pages 723–728.
- Kibriya, A. M., Frank, E., Pfahringer, B., and Holmes, G. (2004). Multinomial Naïve Bayes for Text Categorization Revisited. In *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence, AI'04*, pages 488–499, Berlin, Heidelberg. Springer-Verlag.
- Kjetil Sandve, G. and Drabls, F. (2006). A survey of motif discovery methods in an integrated framework. *Biology Direct*, 1(1).
- Kuksa, P. and Pavlovic, V. (2010). Efficient motif finding algorithms for large-alphabet inputs. *BMC Bioinformatics*, 11(Suppl 8).
- Kulekci, M. O., Vitter, J. S., and Xu, B. (2012). Efficient Maximal Repeat Finding Using the Burrows-Wheeler Transform and Wavelet Tree. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 9(2):421–429.
- Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. (2009). Ultrafast and memory-

- efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):1–10.
- Largeron, C., Moulin, C., and Gèry, M. (2011). Entropy based feature selection for text categorization. In *Proceedings of the 2011 ACM Symp. on Applied Computing, SAC '11*, pages 924–928, New York, NY, USA. ACM.
- Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science (New York, N.Y.)*, 262(5131):208–214.
- Leibovich, L., Paz, I., Yakhini, Z., and Mandel-Gutfreund, Y. (2013). DRIMust: a web server for discovering rank imbalanced motifs using suffix trees. *Nucleic Acids Research*, 41(W1):W174–W179.
- Leslie, C., Eskin, E., and Noble, W. S. S. (2002). The spectrum kernel: a string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pages 564–575, Department of Computer Science, Columbia University, New York, NY 10027, USA. cleslie.noble@cs.columbia.edu.
- Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476.
- Li, G., Chan, T.-M., Leung, K.-S., and Lee, K.-H. (2010). A cluster refinement algorithm for motif discovery. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 7(4):654–668.
- Li, H., Ruan, J., and Durbin, R. (2008a). Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858.
- Li, R., Li, Y., Kristiansen, K., and Wang, J. (2008b). SOAP: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714.

- Li, R., Yu, C., Li, Y., Lam, T.-W., Yiu, S.-M., Kristiansen, K., and Wang, J. (2009). SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967.
- Liao, V.-C. and Chen, M.-S. (2013). Efficient mining gapped sequential patterns for motifs in biological sequences. *BMC Systems Biology*, 7(4):1–13.
- Liao, V.-C. and Chen, M.-S. (2014). DFSP: a Depth-First SPelling algorithm for sequential pattern mining of biological sequences. *Knowledge and Information Systems*, 38(3):623–639.
- Liao, V. C.-C. and Chen, M.-S. (2012). An efficient sequential pattern mining algorithm for motifs with gap constraints. *IEEE International Conference on Bioinformatics and Biomedicine*, 0:1.
- Liu, X. S., Brutlag, D. L., and Liu, J. S. (2002). An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nat Biotech*, 20:835–839.
- Mahmoud, H., Masulli, F., Rovetta, S., and Russo, G. (2014). Community detection in protein-protein interaction networks using spectral and graph approaches. In Formenti, E., Tagliaferri, R., and Wit, E., editors, *Computational Intelligence Methods for Bioinformatics and Biostatistics*, Lecture Notes in Computer Science, pages 62–75. Springer International Publishing.
- Mallek, S., Boukhris, I., and Elouedi, Z. (2015). Predicting proteins functional family: A graph-based similarity derived from community detection. In *Proceedings of the Advances in Intelligent Systems and Computing*, volume 323, pages 629–639. Springer International Publishing.
- Massen, C. P. and Doye, J. P. K. (2005). Identifying communities within energy landscapes. *Phys. Rev. E*, 71:046101.

- Medus, A., Acuna, G., and Dorso, C. (2005). Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2):593–604.
- Melsted, P. and Pritchard, J. (2011). Efficient counting of k-mers in DNA sequences using a Bloom filter. *BMC Bioinformatics*, 12(1):333+.
- Mitchell, M. T. (1997). *Machine learning*. McGraw-Hill Companies Inc., international edition.
- Mora, M., Veggi, D., Santini, L., Pizza, M., and Rappuoli, R. (2003). Reverse vaccinology. *Drug Discov Today*, 8(10):459–464.
- Nakai, K. and Kanehisa, M. (1991). Expert system for predicting protein localization sites in gram-negative bacteria. *Proteins: Structure, Function, and Bioinformatics*, 11(2):95–110.
- Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(026113).
- Ng, H. T., Goh, W. B., and Low, K. L. (1997). Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '97*, pages 67–73, New York, NY, USA. ACM.
- Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management, CIKM '00*, pages 86–93, New York, NY, USA. ACM.
- Pan, Q., Shai, O., Lee, L. J., Frey, B., and Blencowe, B. (2008). Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40:1413–1415.

- Pavesi, G., Mauri, G., and Pesole, G. (2001). An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics (Oxford, England)*, 17 Suppl 1(suppl 1):S207–S214.
- Pavesi, G., Mereghetti, P., Mauri, G., and Pesole, G. (2004). Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res*, 32:199–203.
- Pearson, H. (2006). Genetics: What is a gene? *Nature*, 441(7092):398–401.
- Pennisi, E. (2007). DNA study forces rethink of what it means to be a gene. *Science*, 316(5831):1556–1557.
- Pevzner, P. A. and Sze, S.-H. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press.
- Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663.
- Rafal, P. and Andrzej, P. (2010). BWtrs: A tool for searching for tandem repeats in DNA sequences based on the Burrows Wheeler transform. *Genomics*, 96(5):316 – 321.
- Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106+.
- Rätsch, G., Sonnenburg, S., and Schölkopf, B. (2005). RASE: recognition of alternatively spliced exons in *C.elegans*. *Bioinformatics*, 21(suppl 1):i369–i377.
- Rey, S., Gardy, J., and Brinkman, F. (2005). Assessing the precision of high-throughput computational and laboratory approaches for the genome-wide identification of protein subcellular localization in bacteria. *BMC Genomics*, 6:162.

- Rich, C. and Alexandru, N.-M. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty Third International Conference on Machine Learning (ICML06)*, pages 161–168.
- Rish, I. (2001). An empirical study of the naïve Bayes classifier. In *Workshop on Empirical Methods in AI*.
- Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.
- Roth, F. P., Hughes, J. D., Estep, P. W., and Church, G. M. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nat Biotech*, 16:939–945.
- Saeyns, Y., Rouzè, P., and Van De Peer, Y. (2007). In search of the small ones: improved prediction of short exons in vertebrates, plants, fungi and protists. *Bioinformatics*, 23(4):414–420.
- Sagot, M.-F. (1998). Spelling approximate repeated or common motifs using a suffix tree. In *Proceedings of the Third Latin American Symposium on Theoretical Informatics, LATIN '98*, pages 374–390, London, UK, UK. Springer-Verlag.
- Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.
- Salzberg, S. L., Delcher, A. L., Kasif, S., and White, O. (1998). Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27.

- Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., and Vishwanathan, S. (2009). Hash kernels for structured data. *J. Mach. Learn. Res.*, 10:2615–2637.
- Sinha, S. and Tompa, M. (2003). YMF: a program for discovery of novel transcription factor binding sites by statistical over representation. *Nucleic Acids Research*, 31(13):3586–3588.
- Sun, H., Low, M., Hsu, W., and Rajapakse, J. (2010). RecMotif: a novel fast algorithm for weak motif discovery. *BMC Bioinformatics*, 11(Suppl 11).
- Sun, L., Luo, H., Bu, D., Zhao, G., Yu, K., Zhang, C., Liu, Y., Chen, R., and Zhao, Y. (2013). Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts. *Nucleic Acids Research*.
- Tangirala, K. and Caragea, D. (2014a). Community detection-based features for sequence classification. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB '14*, pages 559–568, New York, NY, USA. ACM.
- Tangirala, K. and Caragea, D. (2014b). Generating features using Burrows Wheeler Transformation for biological sequences. *BIOINFORMATICS*, pages 185–192.
- Tangirala, K. and Caragea, D. (2014c). Semi-supervised classification of protein sequences using burrows wheeler transformation-based features. *6th International Conference on Bioinformatics and Computational Biology (BICoB)*.
- Tangirala, K., Herndon, N., and Caragea, D. (2015). Community detection-based feature construction for protein sequence classification. In *Proceedings of the 11th International Symposium on Bioinformatics Research and Application, ISBRA, 2015*.
- van Laarhoven, T. and Marchiori, E. (2012). Robust community detection methods with resolution parameter for complex detection in protein-protein interaction networks. In

- Shibuya, T., Kashima, H., Sese, J., and Ahmad, S., editors, *Pattern Recognition in Bioinformatics*, volume 7632 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg.
- Vens, C., Rosso, M.-N., and Danchin, E. G. J. (2011). Identifying discriminative classification-based motifs in biological sequences. *Bioinformatics*, 27(9):1231–1238.
- Wang, K., Xu, Y., and Yu, J. X. (2004). Scalable sequential pattern mining for biological sequences. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 178–187, New York, NY, USA. ACM.
- Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA. ACM.
- Wiener, E. D., Pedersen, J. O., and Weigend, A. S. (1995). A neural network approach to topic spotting. In *Proceedings of SDAIR-95*, pages 317–332, Las Vegas, US.
- Wu, X., Zhu, X., He, Y., and Arslan, A. N. (2013). PMBC: Pattern Mining from Biological Sequences with Wildcard Constraints. *Comput. Biol. Med.*, 43(5):481–492.
- Xu, Y., Yang, J., Zhao, Y., and Shang, Y. (2013). An improved voting algorithm for planted (l,d) motif search. *Inf. Sci.*, 237:305–312.
- Yamaguchi, K., Yu, F., and Inoue, M. (1988). Expert system for predicting protein localization sites in gram-negative bacteria. *Cell*, 53(423).
- Yang, Y., Lu, B.-L., and Yang, W.-Y. (2008). Classification of protein sequences based on word segmentation methods. In Brazma, A., Miyano, S., and Akutsu, T., editors, *Proceedings of the Sixth Asia-Pacific Bioinformatics Conference, Kyoto, Japan*, volume 6 of

Advances in Bioinformatics and Computational Biology, pages 177–186. Imperial College Press.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics, ACL '95*, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zaki, M. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1-2):31–60.

Zambelli, F., Pesole, G., and Pavesi, G. (2012). Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in Bioinformatics*.

Zavaljevski, N., Stevens, F. J., and Reifman, J. (2002). Support vector machines with selective kernel scaling for protein classification and identification of key amino acid positions. *Bioinformatics*, 18(5):689–696.

Zhu, F., Yan, X., Han, J., and Yu, P. (2007). Efficient discovery of frequent approximate sequential patterns. In *Seventh IEEE International Conference on Data Mining, ICDM 2007.*, pages 751–756.