DOMAIN ADAPTATION ALGORITHMS FOR BIOLOGICAL SEQUENCE CLASSIFICATION

by

NIC HERNDON

B.S., University of Nevada, Reno, 2004

M.S., University of Nevada, Reno, 2008

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Abstract

The large volume of data generated in the recent years has created opportunities for discoveries in various fields. In biology, next generation sequencing technologies determine faster and cheaper the exact order of nucleotides present within a DNA or RNA fragment. This large volume of data requires the use of automated tools to extract information and generate knowledge. Machine learning classification algorithms provide an automated means to annotate data but require some of these data to be manually labeled by human experts, a process that is costly and time consuming. An alternative to labeling data is to use existing labeled data from a related domain, the source domain, if any such data is available, to train a classifier for the domain of interest, the target domain. However, the classification accuracy usually decreases for the domain of interest as the distance between the source and target domains increases. Another alternative is to label some data and complement it with abundant unlabeled data from the same domain, and train a semi-supervised classifier, although the unlabeled data can mislead such classifier. In this work another alternative is considered, domain adaptation, in which the goal is to train an accurate classifier for a domain with limited labeled data and abundant unlabeled data, the target domain, by leveraging labeled data from a related domain, the source domain. Several domain adaptation classifiers are proposed, derived from a supervised discriminative classifier (logistic regression) or a supervised generative classifier (naïve Bayes), and some of the factors that influence their accuracy are studied: features, data used from the source domain, how to incorporate the unlabeled data, and how to combine all available data. The proposed approaches were evaluated on two biological problems – protein localization and *ab initio* splice

site prediction. The former is motivated by the fact that predicting where a protein is localized provides an indication for its function, whereas the latter is an essential step in gene prediction.

DOMAIN ADAPTATION ALGORITHMS FOR BIOLOGICAL SEQUENCE CLASSIFICATION

by

NIC HERNDON

B.S., University of Nevada, Reno, 2004

M.S., University of Nevada, Reno, 2008

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Doina Caragea

Copyright

Nic Herndon

2016

Abstract

The large volume of data generated in the recent years has created opportunities for discoveries in various fields. In biology, next generation sequencing technologies determine faster and cheaper the exact order of nucleotides present within a DNA or RNA fragment. This large volume of data requires the use of automated tools to extract information and generate knowledge. Machine learning classification algorithms provide an automated means to annotate data but require some of these data to be manually labeled by human experts, a process that is costly and time consuming. An alternative to labeling data is to use existing labeled data from a related domain, the source domain, if any such data is available, to train a classifier for the domain of interest, the target domain. However, the classification accuracy usually decreases for the domain of interest as the distance between the source and target domains increases. Another alternative is to label some data and complement it with abundant unlabeled data from the same domain, and train a semi-supervised classifier, although the unlabeled data can mislead such classifier. In this work another alternative is considered, domain adaptation, in which the goal is to train an accurate classifier for a domain with limited labeled data and abundant unlabeled data, the target domain, by leveraging labeled data from a related domain, the source domain. Several domain adaptation classifiers are proposed, derived from a supervised discriminative classifier (logistic regression) or a supervised generative classifier (naïve Bayes), and some of the factors that influence their accuracy are studied: features, data used from the source domain, how to incorporate the unlabeled data, and how to combine all available data. The proposed approaches were evaluated on two biological problems – protein localization and *ab initio* splice

site prediction. The former is motivated by the fact that predicting where a protein is localized provides an indication for its function, whereas the latter is an essential step in gene prediction.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work was supported by an Institutional Development Award (IDeA) from the National Institute of General Medical Sciences of the National Institutes of Health under grant number P20GM103418. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of General Medical Sciences or the National Institutes of Health.

The computing for this project 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.

I would also like to acknowledge the Department of Computer Science, the Graduate Student Council, and the Engineering Research and Graduate Programs for providing support to attend conferences and present the work in this thesis.

Chapter 1

Introduction

The widespread adoption of next generation sequencing (NGS) technologies enabled faster and cheaper sequencing of DNA and RNA than the previously used Sanger technology, leading to advances in the field of genomics. These technologies generate an abundance of biological data – both raw data, and data derived from primary sequences. In addition, they make it affordable to sequence and analyze new organisms. The analysis of a new organism generally involves three major steps:

1. The first step is to assemble its genome from short DNA read fragments.

2. The second step is to annotate the genome, i.e., to identify the structure and location of the genes. For eukaryotic organisms, accurate gene identification depends heavily on correctly identifying the splice sites (Bernal et al. 2007, Rätsch et al. 2007), the regions of DNA that separate the exons from introns, the donor splice sites, and the introns from exons, the acceptor splice sites. Although the majority of donor and acceptor splice sites, also known as canonical splice sites, are the GT and AG dimers, respectively, only about 1% or less of these two dimers present in a genome are splice sites (Sonnenburg et al. 2007). Untill now, no clear DNA pattern or set of patterns have been identified, either before or after these dimers, that can help in correctly identifying all splice sites, making splice site identification a very difficult task.

3. The third step, after identifying the genes, is to determine the function of the proteins they encode. Considering that the location where a protein localizes is an indicator of its function, protein localization prediction is an important step in determining the function of the proteins.

For the second step, genes identification, a common approach is to assemble short RNA fragments into a transcriptome. The transcriptome is then used as evidence when annotating a genome, by mapping it along that genome. Another option is to map RNA reads along the genome. These approaches help determine the location and structure of the protein-encoding genes. For example, TWINSCAN (Korf et al. 2001) and CONTRAST (Gross et al. 2007) model the entire transcript structure as well as the conserved regions in related species. One of the disadvantages of aligning the transcriptome or RNA reads with the genome to identify the genes is that RNA-Seq reads are generated only from the genes expressed at the time of sample collection in the tissue analyzed, leaving out of the transcriptome some of the protein-encoding genes.

In addition, NGS technologies speed up the sequencing of DNA and RNA molecules, but do so at the expense of read length and accuracy. They generate shorter reads than previous sequencing technologies (e.g., Sanger) with much higher error rates. The common practice to address these issues is to trim the low quality ends of the reads, remove reads with low scores, and require higher depth of coverage. The remaining reads are then assembled into a genome (for DNA reads) or transcriptome (for RNA reads). These assemblies are not 100% accurate. Therefore, annotating a genome using RNA-Seq reads should be validated by independent methods (Steijger et al. 2013).

Machine learning algorithms can be employed to classify biological sequences, especially based on their recent success for many biological problems. Such algorithms could provide not only a cheaper alternative to the more expensive splice site identification with RNA-Seq, but they could potentially also predict splice sites for genes that are not expressed when generating the RNA-Seq reads. Examples of biological problems addressed with machine learning are various. For instance, support vector machines (SVMs) have been used for *ab initio* gene prediction (Bernal et al. 2007), translation initiation identification (Müller et al. 2001, Zien et al. 2000), protein function prediction (Brown et al. 2000), and classification of gene expression profiles into malign and benign (Noble 2006), and hidden Markov models (HMMs) have been used for *ab initio* gene prediction (Hubbard and Park 1995, Stanke and Waack 2003), to name a few.

However, to make accurate predictions, machine learning algorithms need a large amount of labeled data to learn a classifier in a supervised setting. Yet manually labeling enough data for a supervised classifier is costly and time consuming. An option is to learn a classifier from a related organism, assuming that labeled data can be plentifully available for a different, but closely related model organism (for example, a newly sequenced organism is generally scarce in labeled data, whereas a related, well-studied model organism is rich in labeled data). Nevertheless, using a classifier trained on labeled data from the related problem to classify unlabeled data for the problem of interest does not always produce accurate predictions, as the distribution in the source domain is likely different than the distribution in the target domain. Therefore, using supervised machine learning algorithms is not an ideal choice. Another option is to complement the limited labeled data with abundant unlabeled data from the same target domain and learn semi-supervised classifiers. However, the accuracy of such a classifier can be degraded by the unlabeled data (Catal and Diri 2009). A better alternative would be to use domain adaptation algorithms that leverage the large corpus of labeled data from a related, well-studied organism, by combining it with any labeled data and lots of unlabeled data from the organism of interest.

There are challenges with domain adaptation as well, such as:

- Determining what knowledge to transfer from the source domain, and how to transfer this knowledge. Some options include filtering out domain specific features from the target domain, using only instances from the source domain that are highly similar to the instances from the target domain, or a combination of both.

- Deciding whether to incorporate target unlabeled data, as adding unlabeled data could decrease the accuracy of the classifier. In addition, if target unlabeled data is used, how should it be added: iteratively or all at once, with hard labels, soft labels, or a combination of both[1]?

- Identifying the best way to combine all available data: by training a classifier for each dataset and combining their predictions, or by training a classifier on a combination of all data.

In this work several domain adaptation algorithms are proposed, and how the above mentioned factors impact the accuracy of these classifiers are explored.

Published Contributions Included in this Work

The following peer-reviewed publications are included in this work:

1. A Study of Domain Adaptation Classifiers Derived from Logistic Regression for the Task of Splice Site Prediction, (Herndon and Caragea 2016b).

2. *Ab initio* Splice Site Prediction with Simple Domain Adaptation Classifiers, (Herndon and Caragea 2016a).

3. Domain Adaptation with Logistic Regression for the Task of Splice Site Prediction, (Herndon and Caragea 2015a).

4. Empirical Study of Domain Adaptation Algorithms on the Task of Splice Site Prediction, (Herndon and Caragea 2015b).

5. Empirical Study of Domain Adaptation with Naïve Bayes on the Task of Splice Site Prediction, (Herndon and Caragea 2014a).

[1] Soft and hard labels are described in Section 2.1.1

6. Predicting Protein Localization Using a Domain Adaptation Approach, (Herndon and Caragea 2014b).

7. Naïve Bayes Domain Adaptation for Biological Sequences, (Herndon and Caragea 2013).

Chapter 2

Background

2.1 Brief Introduction to Machine Learning

2.1.1 Notation

Let's assume two proteins are given, `MQSARMT` and `MAPYSLL`, and their corresponding locations cytoplasm and inner membrane, respectively[1]. If these proteins are represented as the count of occurrences of each amino-acid, this training data will be:

$$X = \begin{bmatrix} 1 & 0 & 2 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}, Y = \begin{bmatrix} \text{cytoplasm} \\ \text{inner membrane} \end{bmatrix}$$

$m = 2$ is the number of instances, and $n = 9$ is the number of features. The features are `A, L, M, P, Q, R, S, T,` and `Y` (`A` as feature x_1, `L` as feature $x_2, \ldots$ and `T` as feature x_9). Instance $x^1 = \begin{bmatrix} 1 & 0 & 2 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ is the representation of the first protein, `MQSARMT`, as in this protein the amino acid `A` occurs one time ($x_1^1 = 1$), amino acid `L` is not present ($x_2^1 = 0$), amino acid `M` occurs two times ($x_3^1 = 2$), and so on. Its length, $|x^i|$ is 7

[1] This is a purely hypothetical example as in practice the proteins contain longer amino-acid chains, and a machine learning algorithm would use more training instances to build a model.

and its associated label, $y^1 = \text{cytoplasm}$. Throughout this paper the superscript is used to indicate the instance number and the subscript to indicate the feature number.

There are two types of labels assigned to unlabeled data:

- *Soft* labels means that if for an instance x^i a classifier predicts that $P(y^i = 1 \mid x^i) = 0.8$ and $P(y^i = 0 \mid x^i) = 0.2$, then the instance is labeled it with $y^i = (0.8, 0.2)$.
- *Hard* labels means that if for an instance x^i a classifier predicts $P(y^i = 1 \mid x^i) = 0.8$ and $P(y^i = 0 \mid x^i) = 0.2$, then the instance is labeled with $y^i = (1, 0)$.

2.1.2 Supervised Classification

A supervised machine learning algorithm takes a set of training instances $X \in \mathbb{R}^{m \times n}$, where m is the number of instances and n is the number of features, and their corresponding labels $Y \in \mathcal{Y}^m$ to generate a model. Then, given a new instance x^i, this classifier[2] will predict the label for this instance.

2.1.3 Semi-supervised Classification

A semi-supervised machine learning algorithm takes a set of training instances $X_L \in \mathbb{R}^{m_L \times n}$ with their corresponding labels $Y_L \in \mathcal{Y}^{m_L}$, and a set of unlabeled instances $X_U \in \mathbb{R}^{m_U \times n}$, and uses them to generate a model. Then, given a new instance, this classifier will predict the label for this instance.

2.1.4 Supervised Domain Adaptation Classification

A supervised domain adaptation machine learning algorithm takes a set of labeled instances from a domain of interest, the target domain, $X_{tTL} \in \mathbb{R}^{m_{tTL} \times n}$ with their corresponding labels $Y_{tTL} \in \mathcal{Y}^{m_{tTL}}$, and a set of training instances from a related domain, the source

[2] Throughout this paper the terms classifier and machine learning algorithm are used interchangeably.

domain, $X_{tSL} \in \mathbb{R}^{m_{tSL} \times n}$ with their corresponding labels $Y_{tSL} \in \mathcal{Y}^{m_{tSL}}$, and uses them to generate a model for the target domain. Then, given a new instance from the target domain, this classifier will predict the label for this instance.

2.1.5 Semi-supervised Domain Adaptation Classification

A semi-supervised domain adaptation machine learning algorithm takes a set of labeled instances from a domain of interest, the target domain, $X_{tTL} \in \mathbb{R}^{m_{tTL} \times n}$ with their corresponding labels $Y_{tTL} \in \mathcal{Y}^{m_{tTL}}$, a set of unlabeled instances from the target domain, $X_{tTU} \in \mathbb{R}^{m_{tTU} \times n}$, and a set of training instances from a related domain, the source domain, $X_{tSL} \in \mathbb{R}^{m_{tSL} \times n}$ with their corresponding labels $Y_{tSL} \in \mathcal{Y}^{m_{tSL}}$, and uses them to generate a model for the target domain. Then, given a new instance from the target domain, this classifier will predict the label for this instance.

2.2 Central Dogma of Molecular Biology

The blueprint for any living organism is contained within its chromosome or chromosomes, which are long molecules of deoxyribonucleic acid (DNA). The DNA has regions that encode proteins – the genes. In eukaryotic organisms – the organisms with cells containing a nucleus and other organelles – the genes contain encoding regions, or exons, separated by non-encoding regions, or introns, as shown in Figure 2.1[3]. There are other regions within a gene, such as the promoter region and untranslated regions, but these are beyond the scope of this work.

The introns are removed, or spliced out, after which adjacent exons are concatenated and then transcribed into messenger ribonucleic acid (mRNA). The mRNA then exits the cell nucleus where the DNA is housed, and enters the cell's cytoplasm, where it is translated into

[3] Image by BCSteve - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=30096313 downloaded on March 27, 2016

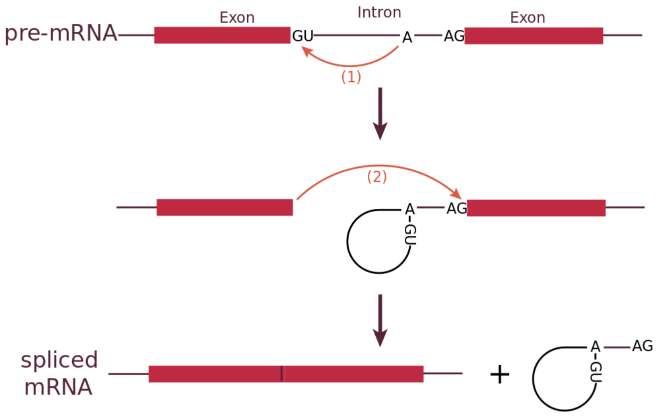


Figure 2.1: RNA splicing.

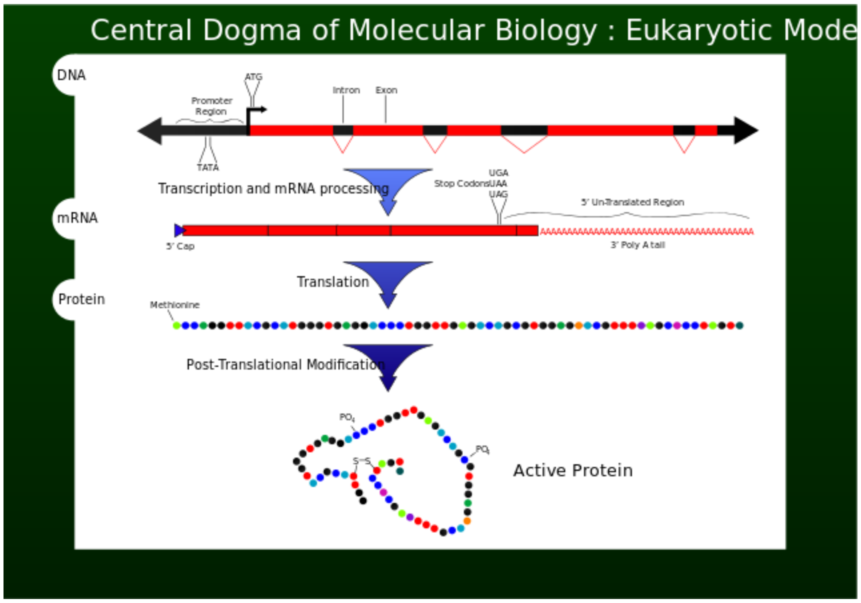


Figure 2.2: Central dogma of molecular biology.

amino-acids that are chained and folded to form proteins. This flow of genetic information is known as the central dogma of molecular biology and is shown in Figure 2.2[4].

In most cases, the transition from exon to intron occurs at `GT` dimer, called the donor spice site, and the transition from intron to exon occurs at the `AG` dimer, called the acceptor

[4] Image by Adenosine at English Wikipedia, CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=32026515 downloaded on April 16, 2016

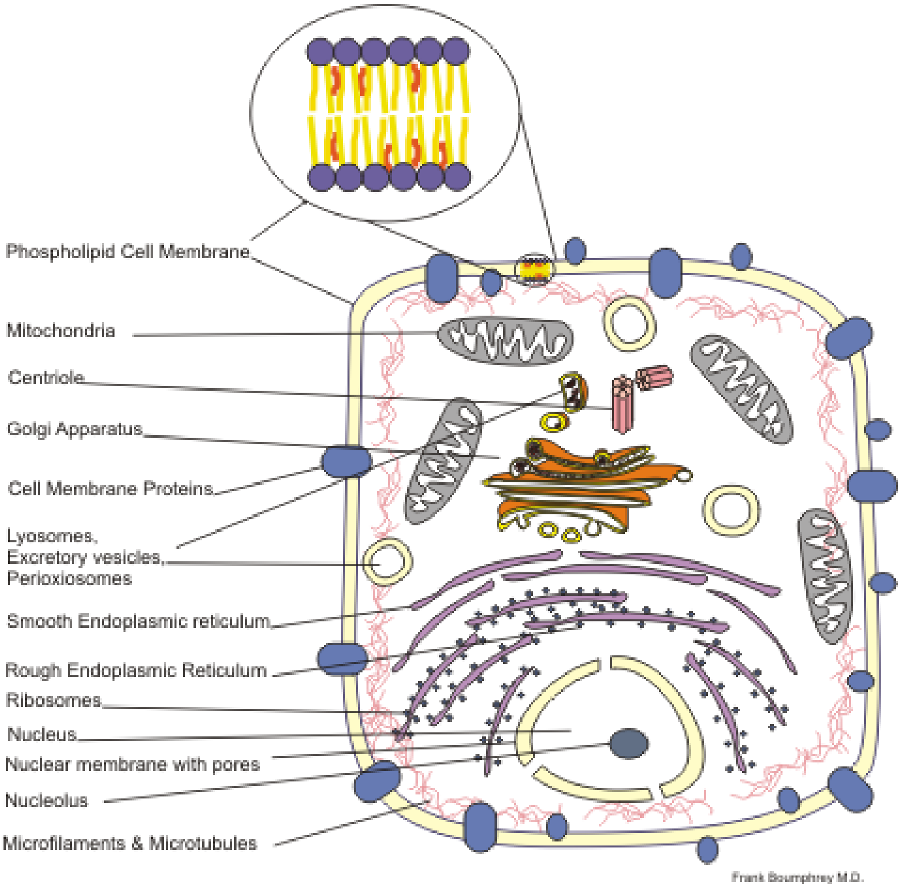


Figure 2.3: Cell structure showing different localizations of proteins.

splice site.

2.3 Protein Localization

One of the goals of analyzing proteins is to determine their function. One indicator of a protein's function is the cellular localization site of the protein, such as periplasm or the extracellular environment, as shown in Figure 2.3[5]. Determining the localization of a protein through experiments is time consuming and laborious, but can also be determined from the amino acid sequence of the protein, using computational tools.

[5] Image by Boumphreyfr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6882429 downloaded on April 3, 2016

Chapter 3

Related Work

The following sections present previous related work. Section 3.1 presents machine learning approaches for splice site prediction, Section 3.2 presents computational tools for protein localization, and the following sections present machine learning methods used when there is a limited amount of labeled data: Section 3.3 presents semi-supervised classifiers, Section 3.4 presents domain adaptation classifiers, and Section 3.5 presents different ways to incorporate unlabeled data.

3.1 Splice Site Prediction

Most of the approaches addressing splice site prediction involve supervised learning. For example, Li et al. (2012) proposed a method that used the discriminating power of each position in the DNA sequence around the splice site, estimated using the chi-square test. They used a support vector machine algorithm with a radial basis function kernel that combines the scaled component features, the nucleotide frequencies at conserved sites, and the correlative information of two sites, to train a classifier for the human genome. Baten et al. (2006), Sonnenburg et al. (2007), and Zhang et al. (2006), also proposed supervised support vector machine classifiers, whereas Baten et al. (2007) proposed a method using a hidden Markov model, Cai et al. (2000) proposed a Bayesian network algorithm, and Arita et al. (2002) proposed a method using Bahadur expansion truncated at the second order. For more work on gene prediction using supervised learning, see the survey by Al-Turaiki et al. (2011). However, one major drawback of these supervised algorithms is that they typically require large amounts of labeled data to train a classifier. There are also evidence-based methods, such as TWINSCAN (Korf et al. 2001), CONTRAST (Gross et al. 2007), TrueSight (Li et al. 2013), and using single-molecule transcript sequencing (Minoche et al. 2015). It is however unfair to compare these with *ab initio* methods, as they use mRNA evidence to generate their models, whereas *ab initio* methods do not.

3.2 Protein Localization

Numerous computational methods for predicting protein localization are available (for a review see (Emanuelsson 2002)). PSORTb (Gardy et al. 2003) is one of the first widely used method. It uses a Bayesian network to combine the output of several modules – homology analysis, motif-based analysis, detection of transmembrane alpha-helices, outer membrane protein motif analysis, signal peptide predictor, and amino acid composition analysis using SVM – to generate protein localization predictions. Although the classification precision was high, its predictive coverage was low and only applicable to Gram-negative bacteria. An updated version, PSORTb v.2.0 (Gardy et al. 2005), increased the previous version's coverage and expanded it to include Gram-positive bacteria. It also uses a Bayesian network to combine the output of several modules. The amino acid composition analysis module was updated to use a new SVM-based method, the signal peptide predictor was trained with Gram-positive and Gram-negative data, and the homology and motif modules searched against expanded databases. Another method, TargetP (Emanuelsson et al. 2000), trains a neural network using only the N-terminal sequence information to discriminate between proteins.

3.3 Semi-supervised Classifiers

An alternative, when the amount of labeled data is not enough for learning a supervised classifier, is to use the limited amount of labeled data in conjunction with abundant unlabeled data to learn a semi-supervised classifier. For example, Nigam et al. (2000) showed empirically that combining a small labeled dataset with a large unlabeled dataset from the same or different domains can reduce the classification error of text documents by up to 30%. Their algorithm uses a combination of Expectation Maximization and the Naïve Bayes classifier by first learning a classifier on the labeled data which is then used to classify the unlabeled data. The combination of these datasets trains a new classifier and iterates until convergence. By incorporating unlabeled data, a semi-supervised classifier requires less labeled data than a supervised classifier, to learn an accurate model.

However, semi-supervised classifiers could be misled by the unlabeled data, especially when there is hardly any labeled data (Catal and Diri 2009). For example, if during the first iteration one or more instances are misclassified, the semi-supervised algorithm will be skewed towards the mislabeled instances in subsequent iterations. Another deficiency of semi-supervised classifiers is that their accuracy decreases as the imbalance between classes increases. This is a major challenge for the task of splice site as the classes are highly imbalanced, with only about one percent positive instances. Note that these two challenges affect other algorithms (e.g., domain adaptation), and the data imbalance challenge is common to other problems as well, such as intrusion detection, medical diagnosis, risk management, and text classification, to name a few. The proposed solutions address this problem at the algorithmic level or at the data level through resampling. For an overview of solutions to imbalanced data sets see (Chawla et al. 2004, He and Garcia 2009). For splice site prediction, Stanescu and Caragea (2014a) studied the effects of imbalanced data on semi-supervised algorithms and found that although self-training that adds only positive instances in the semi-supervised iterations achieved the best results out of the methods evaluated, oversampling and ensemble learning are better options when the positive-to-negative ratio is about 1:99. In their subsequent study (Stanescu and Caragea 2014b), they evaluated several ensemble-based semi-supervised learning approaches, out of which, again, a self-training ensemble with only positive instances produced the best results. However, the highest area under precision-recall curve for the best classifier was 54.78%.

3.4 Domain Adaptation Classifiers

Another option that addresses the lack of abundant labeled data needed with supervised algorithms is to use domain adaptation. This approach has been successfully applied to other problems even when the base learning algorithms used in domain adaptation make simplifying assumptions. For instance, in text classification, Dai et al. (2007) proposed an iterative algorithm derived from naïve Bayes that uses expectation-maximization for classifying text documents into top categories. This algorithm performed better than supervised SVM and naïve Bayes classifiers when tested on datasets from Newsgroups, SRAA and Reuters. A similar domain adaptation algorithm derived from the Naïve Bayes classifier is the Adapted Naïve Bayes classifier (Tan et al. 2009), which identifies and uses only the generalizable features from the source domain, and the unlabeled data with all the features from the target domain to build a classifier for the target domain. This algorithm was evaluated on the task of sentiment analysis. The prediction rate was promising, with Micro F1 values between 0.69 and 0.90, and Macro F1 values between 0.59 and 0.91. However, the classifier did not use any labeled data from the target domain. For more work on domain adaptation and transfer learning, see the survey by Pan and Yang (2010).

Even though domain adaptation has been used with good results in other domains, there are only a few domain adaptation methods proposed for biological problems. In a recent approach for splice site prediction, Giannoulis et al. (2014) proposed a modified version of the k-means clustering algorithm that took into account the commonalities between the source and target domains for splice site prediction. This algorithm was not very accurate though. Its best area under receiver operating characteristic curve (auROC) was below 70%. The best results for the task of splice site prediction, especially when the source and target domains were not closely related, were obtained with a dual-task learning support vector machine classifier proposed by Schweikert et al. (2008), $\mathrm{SVM}_{\mathrm{S,T}}$, which used a weighted degree kernel proposed by Rätsch et al. (2007). Schweikert et al. (2008) used the kernel to generate values between 0 and 1 by normalizing the count of identical dimers at the same position within two DNA fragments of length 141. With this kernel, they solved both classification problems concurrently, for the source and for the target domains, by coupling their solutions via a regularization term. This classifier though did not utilize the abundant unlabeled data from the target domain.

3.5 Usage Styles for Unlabeled Data

If the abundant unlabeled data from the target domain is incorporated one needs to explore the two main methods of using unlabeled data when training a classifier: by assigning hard labels with self-training, or by assigning soft labels with an expectation-maximization (EM) algorithm.

The self-training algorithm (Maeireizo et al. 2004, Riloff et al. 2003, Yarowsky 1995) is an iterative method of using the unlabeled data, that first learns a classifier from only the labeled data. This classifier is then used on the unlabeled data to generate more hard-labeled examples by selecting the instances most confidently classified and assigning labels to them. These instances are then moved from the unlabeled data to the labeled data, and this process is repeated until the classifier converges or a set number of iterations has been reached. This approach has been effective for diverse applications, such as text classification (Blum and Mitchell 1998), optical character recognition (Zhu and Ghahramani 2002), and face recognition (Roli and Marcialis 2006), to name a few.

The expectation-maximization (EM) algorithm (Dempster et al. 1977) is another iterative method for using unlabeled data. In the expectation step, the likelihood is evaluated using the current parameters, whereas in the maximization step the parameters are computed by maximizing the expected likelihood evaluated in the expectation step. The label assigned to each instance from the unlabeled data set, that maximizes this expected likelihood, is a soft label. Similar to the self-training algorithm, this process is repeated until convergence or until reaching the maximum number of iterations.

The EM algorithm was implemented for diverse applications, such as text classification, and biological sequences classification, in domain adaptation settings. For instance, in text classification, the Naïve Bayes Transfer Classification algorithm (Dai et al. 2007), assumes that the source and target data have different distributions. It trains a classifier on source data and then applies the EM algorithm to fit the classifier for the target data, using the Kullback-Liebler divergence to determine the trade-off parameters in the EM algorithm. When tested on datasets from Newsgroups, SRAA and Reuters for the task of top-category classification of text documents this algorithm performed better than support vector machine and Naïve Bayes classifiers.

Another study of EM for text classification (Nigam et al. 2000) used a combination of EM and the Naïve Bayes classifier by first learning a classifier on the labeled data which is then used to classify the unlabeled data. The combination of these datasets trained a new classifier and iterated until convergence. By augmenting the labeled data with unlabeled data the classifier required less labeled data as compared to using only labeled data with a supervised classifier. This algorithm reduced the classification error of text documents by up to 30%.

Both of these methods make one or more of the following assumptions when using the unlabeled data. The self-training algorithm assumes that: instances that are close to each other in the hyperspace have the same label (the smoothness assumption), instances form discrete clusters with instances in a cluster having the same label (the cluster assumption), and instances can be represented in a lower dimensional space than the input space without significant loss of information (the manifold assumption). The expectation-maximization algorithm makes the assumption that there are features with missing values or that a simpler model can be learned by using additional unobserved instances.

Therefore, when it cannot be determined *a priori* if these assumptions are valid for the data set used, one should evaluate both methods, and the combination of both, to determine which one works best for the data set and the algorithm used.

Chapter 4

Supervised Classifiers

Several domain adaptation methods were proposed, derived from the naïve Bayes, and the regularized logistic regression supervised classifiers. Before describing the proposed methods the supervised classifiers are presented, as these are used as baselines, and to highlight the differences between the proposed methods and these classifiers.

4.1 Multinomial Naïve Bayes

The multinomial naïve Bayes classifier (McCallum et al. 1998) assumes that the sample data used to train the classifier is representative of the population data on which the classifier will be used. In addition, it assumes that the frequency of the words determines the label assigned to an instance, and that the position of a word is irrelevant (the naïve Bayes assumption). Thus, using Bayes' property a classifier can approximate the posterior probability, i.e., the probability of a class given an unclassified instance, as being proportional to the product of the prior probability of the class, and the probability of the instance given the class:

$$P(y^i = y \mid x^i) \propto P(y^i = y)P(x^i \mid y^i = y) \tag{4.1}$$

where the probability of the class is

$$P(y^i = y) = \frac{\sum_{i=1}^{m} \mathbb{1}(y^i = y)}{m} \tag{4.2}$$

with $\mathbb{1}(y^i = y) = 1$ if $y^i = y$, and 0 otherwise.

The probability of an instance given its class is the multinomial distribution:

$$P(x^i \mid y^i = y) = P(|x^i|)|x^i|! \prod_{j=1}^{n} \frac{P(x_j \mid y^i = y)^{x^i_j}}{x^i_j!} \tag{4.3}$$

where

$$P(x_j \mid y^i = y) = \frac{1 + \sum_{i=1}^{m} x^i_j \mathbb{1}(y^i = y)}{n + \sum_{k=1}^{n} \sum_{i=1}^{m} x^i_k \mathbb{1}(y^i = y)}$$

and

$$P(|x^i|) = P(|x^i| = l) = \frac{\sum_{i=1}^{m} \mathbb{1}(|x^i| = l)}{m}$$

The document length is included in Equation (4.4) because it specifies the number of draws from the multinomial, with the assumption that the document length is not dependent on class.

4.2 Naïve Bayes

Similar to the multinomial naïve Bayes classifier, the multivariate Bernoulli naïve Bayes classifier assumes that the position of a feature is irrelevant (the naïve Bayes assumption), though, unlike the multinomial naïve Bayes classifier, it represents each instance as a vector of binary features indicating whether the corresponding feature occurs in the instance or not. It also uses Bayes' property to approximate the posterior probability, using Equation (4.1).

The probability of an instance given its class is the product of probabilities of all feature values, including the probability of non-occurrence for features that do not occur in the instance:

$$P(x^i \mid y^i = y) = \prod_{j=1}^{n} \left\{ x_j^i P(x_j \mid y^i = y) + (1 - x_j^i) \left[1 - P(x_j \mid y^i = y) \right] \right\} \tag{4.4}$$

where

$$P(x_j \mid y^i = y) = \frac{1 + \sum_{i=1}^{m} x_j^i \mathbb{1}(y^i = y)}{2 + \sum_{i=1}^{m} \mathbb{1}(y^i = y)}$$

4.3 Regularized Logistic Regression

Given a set of training instances generated independently $X \in \mathbb{R}^{m \times n}$ and their corresponding labels $Y \in \mathcal{Y}^m$, $\mathcal{Y} = \{0, 1\}$, with m the number of training instances and n the number of features, logistic regression models the posterior as (Le Cessie and Van Houwelingen 1992):

$$\begin{aligned} P(y^i = y \mid x^i; \theta) &= \begin{cases} g(\theta^T x^i) & \text{, if } y = 1 \\ 1 - g(\theta^T x^i) & \text{, if } y = 0 \end{cases} \\ &= \left[g(\theta^T x^i) \right]^y \cdot \left[1 - g(\theta^T x^i) \right]^{1-y} \end{aligned}$$

where $g(\cdot)$ is the logistic function $g(\theta^T x^i) = \frac{1}{1+e^{-\theta^T x^i}}$.

With this model, the log likelihood can be written as a function of the parameters θ as follows:

$$\begin{aligned} l(\theta) &= \log \prod_{i=1}^{m} P(y \mid x^i; \theta) \\ &= \log \prod_{i=1}^{m} \left[g(\theta^T x^i) \right]^y \cdot \left[1 - g(\theta^T x^i) \right]^{1-y} \\ &= \sum_{i=1}^{m} \left[y \log g(\theta^T x^i) + (1 - y) \log \left(1 - g(\theta^T x^i) \right) \right] \end{aligned}$$

The parameters are estimated by maximizing the log likelihood, usually using maximum entropy models, after a regularization term, with parameter λ, is introduced to penalize large values of θ:

$$\theta = \arg\max_{\theta} \left[l(\theta) - \lambda \|\theta\|^2 \right] \tag{4.5}$$

Note that x^i is the i^{th} sequence in the training data set, y^i is the corresponding label of x^i, and $x_0^i = 1, \forall i \in \{1, 2, \ldots, m\}$ such that $\theta^T x^i = \theta_0 + \sum_{j=1}^{n} \theta_j x_j^i$.

Chapter 5

Domain Adaptation Classifiers

One limitation of the supervised classifiers is that when trained on one domain and then used on a different domain, in most cases, their classification accuracy decreases. The first method proposed to address this issue, used the Adapted Naïve Bayes classifier proposed by Tan et al. (2009), with two modifications: the labeled data from the target domain was used, and the self-training technique to assign labels to instances from target unlabeled dataset was employed. These modifications will be described in more detail shortly. The second method proposed further modified this algorithm: normalized the counts used in computing the prior and the likelihood, used mutual information instead of probabilities when ranking the source domain features, and used different representation for the input data. The third method proposed is derived from a discriminative classifier instead of generative one, namely, regularized logistic regression. The last method proposed assigned different weights to the labeled data from the source and target domains, then trained a supervised classifier on the combined dataset.

5.1 Semi-supervised Domain Adaptation Classifier Derived from Multinomial Naïve Bayes with Counts of k-mer Features

The first step of this classifier is to identify in the source domain the subset of the features that generalize well and are highly correlated with the label. Then, use the data from the source domain with only these features and the data from the target domain with all the features to predict the labels of the test instances in the target domain. Theoretically, the set of features in each domain can be split into four categories, based on two selection criteria. Based on the correlation between the feature and the label, the features can be divided into features that are highly related to the labels, and features that are less related to the labels. Based on the specificity of the features, the features can be divided into features that are very specific to a domain, and features that generalize well across related domains, as shown in Figure 5.1.

To select informative features from the source domain the features are ranked based on their probabilities. The features that are generalizable between source and target domains would most likely occur frequently in both domains, and should be ranked higher. In addition, the features that are correlated to the labels should also be ranked higher. Therefore,

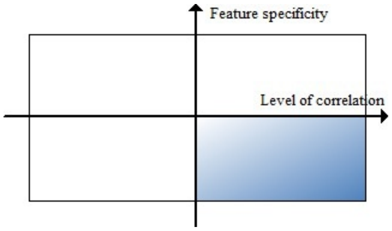


Figure 5.1: The features of interest are generalizable and highly correlated to class, i.e., the ones in the highlighted quadrant.

the following measure is used to rank the features in the source domain:

$$f(x_j) = \log \frac{P_{tSL}(x_j) \cdot P_{tTL}(x_j)}{|P_{tSL}(x_j) - P_{tTL}(x_j)| + \alpha} \tag{5.1}$$

where P_{tSL} and P_{tTL} are the probability of the feature $x_j, \forall j \in \{1, \ldots, n\}$ in the source and target domain, respectively. The numerator ranks higher the features that occur frequently in both domains, since the larger both probabilities are the larger the numerator is, and thus the higher the rank of the feature is. The denominator ranks higher the features that have similar probabilities (i.e., the generalizable features), since the closer the probabilities are for a feature in both domains, the smaller the denominator value is, and thus the higher the rank. The additional value in the denominator, α, is used to prevent division by zero. The higher its value is the more influence the numerator has in ranking the features, and vice versa. To limit its influence on ranking the features, a small value was chosen for this parameter, 0.0001. The probability of a feature in either domain is

$$P(x_j) = \frac{\displaystyle\sum_{i=1}^{m} x_j^i + \beta}{\displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} x_j^i + m \cdot \beta} \tag{5.2}$$

where β is a smoothing factor, which is used to prevent the probability of a feature to be 0 (which would make the numerator in Equation (5.1) equal to 0, and the logarithm function is undefined for 0). A small value was chosen for β as well, 0.0001, to limit its influence on the ranking of features. Note that the values for α and β do not have to be the same, but they can be, as used by Tan et al. (2009) and in these experiments.

Once the domain specific features of the source domain are filtered out, the algorithm uses a combination of the expectation-maximization (*EM*) algorithm and a weighted multinomial naïve Bayes algorithm. Similar to the *EM* algorithm, it has two steps that are iterated until convergence. The first step, the *M*-step, simultaneously estimates the class probability and the class conditional probability of a feature. However, unlike the *EM* algorithm that uses the data from one domain to calculate these values, this algorithm uses a weighted combination of the data from the source domain and the target domain.

$$P(y^i = y) = \frac{(1-\lambda)\sum_{i=1}^{m_{tSL}} \mathbb{1}(y^i = y) + \lambda \sum_{i=1}^{m_{tTL}} \mathbb{1}(y^i = y)}{(1-\lambda)m_{tSL} + \lambda m_{tTL}} \tag{5.3}$$

$$P(x_j \mid y^i = y) = \frac{(1-\lambda)\sum_{i=1}^{m_{tSL}} \eta_j x_j^i \mathbb{1}(y^i = y) + \lambda \sum_{i=1}^{m_{tTL}} x_j^i \mathbb{1}(y^i = y) + 1}{(1-\lambda)\sum_{k=1}^{n}\sum_{i=1}^{m_{tSL}} \eta_k x_k^i \mathbb{1}(y^i = y) + \lambda \sum_{k=1}^{n}\sum_{i=1}^{m_{tTL}} x_k^i \mathbb{1}(y^i = y) + n} \tag{5.4}$$

where λ is the weight factor between the source and target domains:

$$\lambda = \min\{\delta \cdot \tau, 1\} \tag{5.5}$$

and τ is the iteration number. $\delta \in (0, 1)$ is a constant that determines how fast the weight shifts from the source domain to the target domain, and η_j is 1 if feature x_j in the source domain is a generalizable feature, 0 otherwise.

Unlike the algorithm proposed by Tan et al. (2009), which considers that all the instances from the target domain are unlabeled and does not use them during the first iteration (i.e., $\lambda = 0$), it is reasonable to assume that there is a small number of labeled instances in the target domain, and the proposed algorithm uses any labeled data from the target domain in the first and subsequent iterations. In the first iteration only labeled instances from the source and target domains are used to estimate the probability distributions for the class conditional probabilities given the instance. In subsequent iterations the class of the instance for the labeled data from the source and target domains and the probability distribution of the class for the unlabeled data from the target domain are used.

The second step, the *E*-step, estimates the probability of the class for each instance with the values obtained from the M-step.

$$P(y^i = y \mid x^i) \propto P(y^i = y)P(|x^i|)|x^i|! \prod_{j=1}^{n} \frac{P(x_j \mid y^i = y)^{x_j^i}}{x_j^i!} \tag{5.6}$$

The second modification made to the classifier proposed by Tan et al. (2009), is the use of self-training, i.e., at each iteration, the instances with the top class probability are selected, proportional to the class distribution, and considered to be labeled in the subsequent iterations. This improves the prediction accuracy of the classifier because it does not allow the unlabeled data to alter the class distribution from the target labeled data.

The two steps, E and M, are repeated until the instance conditional probabilities values in Equation (5.6) converge (or a given number of iterations is reached). The algorithm is summarized in Algorithm 1, and shown in Figure 5.2.

Algorithm 1 Outline of the semi-supervised domain adaptation classifier derived from multinomial naïve Bayes, SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$.

```
1: Select generalizable features from the source domain, i.e., the top ranked features using
   Equation (5.1).
2: For each class simultaneously estimate the class probability and the class conditional
   probability of each feature using Equations (5.3) and (5.4), respectively. For the source
   domain use all labeled instances, and only the generalizable features. For the target
   domain use only labeled instances, and all features.
3: Self-training: Select, proportional to the prior class distribution, the target instances
   with the top class probability, and consider these to be labeled (i.e., assign them hard-
   labels) in the subsequent iterations; assign soft-labels to the remaining instances.
4: while labels assigned to unlabeled data change do
5:     $M$-step: Same as step 2 but use the class for labeled and self-trained instances from
       the target domain, and the class distribution for unlabeled instances.
6:     Same as step 3.
7:     $E$-step: Estimate the class distribution for unlabeled training instances from the
       target domain using Equation (5.6).
8: end while
9: Use classifier to label new target data.
```

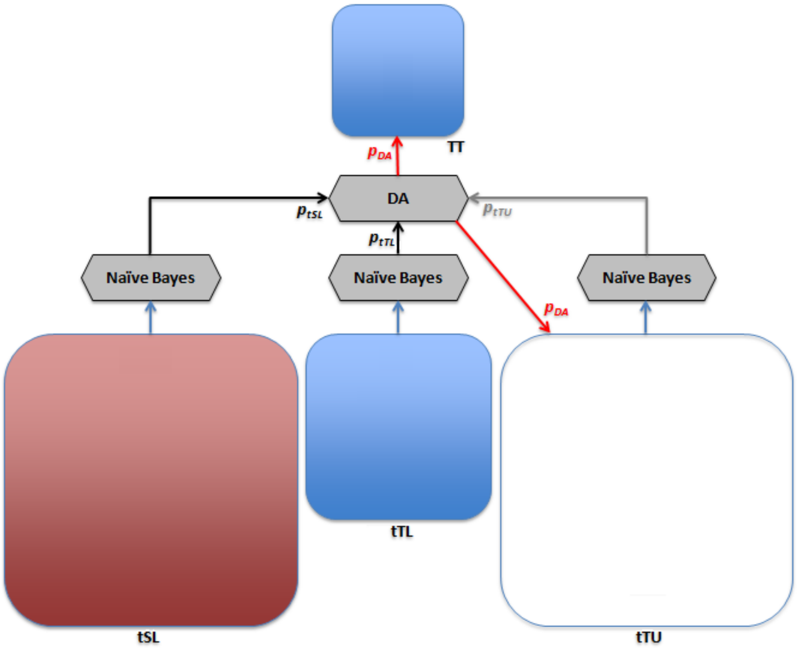


Figure 5.2: Semi-supervised domain adaptation classifier derived from multinomial naïve Bayes.

5.1.1 Data Sets

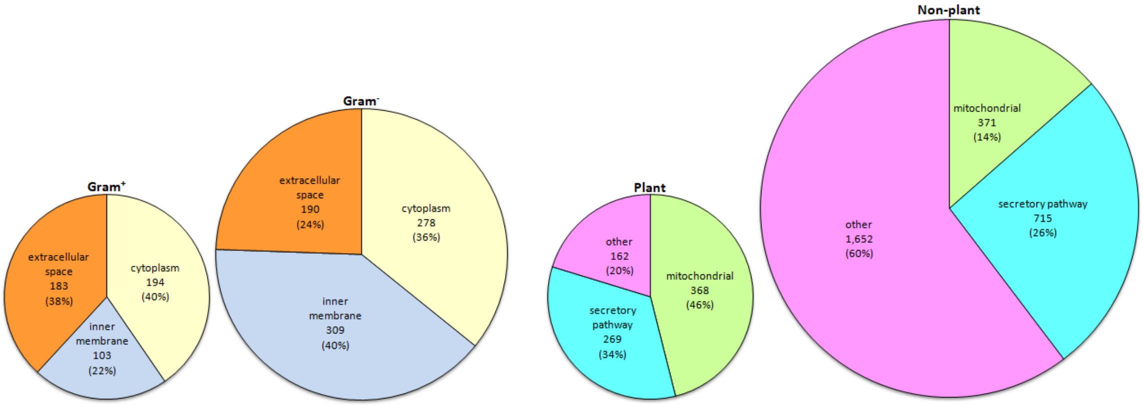
Three data sets from two biological problems – protein localization, and splice site prediction – were used to evaluate this and the other proposed methods.

Protein Localization

For protein localization the following two datasets were used:

- The PSORTb v2.0[1] dataset (Gardy et al. 2005), first introduced in (Gardy et al. 2003). It contains proteins from two related prokaryotes, gram-negative and gram-positive bacteria, and their primary localization: cytoplasm, inner membrane, periplasm, outer membrane, and extracellular space. Only instances with classes that appear in both datasets were used: 480 proteins from gram-positive bacteria (194 from cytoplasm, 103 from inner membrane, and 183 from extracellular space) and 777 proteins from

[1] Downloaded from http://www.psort.org/dataset/datasetv2.html



(a) Prokaryotes

(b) Eukaryotes

Figure 5.3: Number of proteins used from each dataset. From the PSORTb v2.0 dataset 194 proteins localized in cytoplasm, 103 in inner membrane, and 183 in extracellular space, were used for gram-positive, and 278 proteins localized in cytoplasm, 309 in inner membrane, and 190 in extracellular space, for gram-negative. From the TargetP dataset 368 mitochondrial, 269 secretory pathway, and 162 "other" proteins from plant were used, and 371 mitochondrial, 715 secretory pathway, and 1,652 "other" proteins from non-plant.

gram-negative bacteria (278 from cytoplasm, 309 from inner membrane, and 190 from extracellular space), as shown in Figure 5.3a.

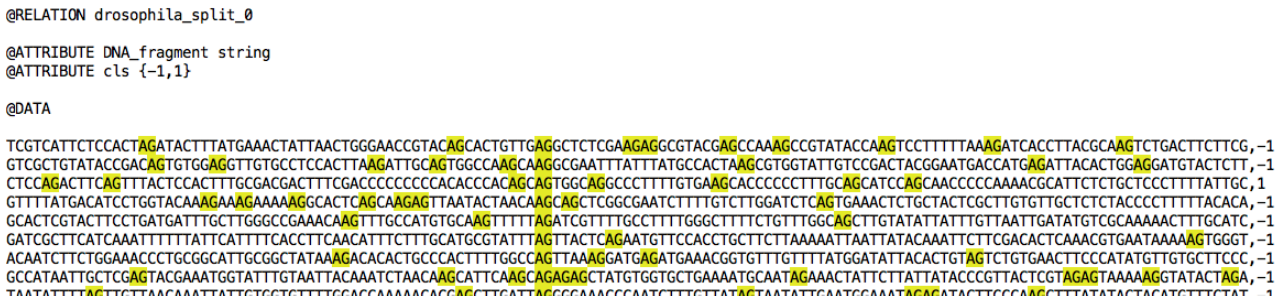
- The TargetP[2] dataset, first introduced in (Emanuelsson et al. 2000). It contains proteins from two distantly related organisms, plant and non-plant, and their primary sub-cellular localization: mitochondrial, chloroplast, secretory pathway, and "other." Similar to the bacteria proteins, from this data set 799 plant proteins were used (368 mitochondrial, 269 secretory pathway and 162 "other") and 2,738 non-plant proteins (371 mitochondrial, 715 secretory pathway and 1652 "other"), as shown in Figure 5.3b.

Splice Site

For splice site prediction the dataset[3] first used in (Schweikert et al. 2008) was used, which contains sets of 141 nucleotides-long DNA sequences from five organisms. Each sequence has

[2] Downloaded from http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php

[3] Downloaded from http://ftp.raetschlab.org/user/cwidmer



```
@RELATION drosophila_split_0

@ATTRIBUTE DNA_fragment string
@ATTRIBUTE cls {-1,1}

@DATA

TCGTCATTCTCCACTAGATACTTTATGAAACTATTAACTGGGAACCGTACAGCACTGTTGAGGCTCTCGAAGAGGCGTACGAGCCAAAGCCGTATACCAAGTCCTTTTTAAAGATCACCTTACGCAAGTCTGACTTCTTCG,-1
GTCGCTGTATACCGACAGTGTGGAGGTTGTGCCTCCACTTAAGATTGCAGTGGCCAAGCAAGGCGAATTTATTTATGCCACTAAGCGTGGTATTGTCCGACTACGGAATGACCATGAGATTACACTGGAGGATGTACTCTT,-1
CTCCAGACTTCAGTTTACTCCACTTTGCGACGACTTTCGACCCCCCCCACACCCACAGCAGTGGCAGGCCCTTTTGTGAAGCACCCCCCTTTGCAGCATCCAGCAACCCCCAAAACGCATTCTCTGCTCCCTTTTATTGC,1
GTTTTATGACATCCTGGTACAAAGAAAGAAAAAGGCACTCAGCAAGAGTTAATACTAACAAGCAGCTCGGCGAATCTTTTGTCTTGGATCTCAGTGAAACTCTGCTACTCGCTTGTGTTGCTCTCTACCCCTTTTTACACA,-1
GCACTCGTACTTCCTGATGATTTGCTTGGGCCGAAACAAGTTTGCCATGTGCAAGTTTTTAGATCGTTTTGCCTTTTGGGCTTTTCTGTTTGGCAGCTTGTATATTATTTGTTAATTGATATGTCGCAAAAACTTTGCATC,-1
GATCGCTTCATCAAATTTTTTATTCATTTTCACCTTCAACATTTCTTTGCATGCGTATTTAGTTACTCAGAATGTTCCACCTGCTTCTTAAAAATTAATTATACAAATTCTTCGACACTCAAACGTGAATAAAAAGTGGGT,-1
ACAATCTTCTGGAAACCCTGCGGCATTGCGGCTATAAAGACACACTGCCCACTTTTGGCCAGTTAAAGGATGAGATGAAACGGTGTTTGTTTTATGGATATTACACTGTAGTCTGTGAACTTCCCATATGTTGTGCTTCCC,-1
GCCATAATTGCTCGAGTACGAAATGGTATTTGTAATTACAAATCTAACAAGCATTCAAGCAGAGAGCTATGTGGTGCTGAAAATGCAATAGAAACTATTCTTATTATACCCGTTACTCGTAGAGTAAAAAGGTATACTAGA,-1
TAATATTTTAGTTCTTAACAAATTATTCTGCTGTTTTGCACCAAAAACACCAGCTTGATTAGCGGAAACCCAATCTTTCTTATAGTAATATTGAATGGAAATAGAGATACTTCCCAAGCTTTATATACTACATGTTTCTAT,-1
```

Figure 5.4: DNA fragments of 141 bp and their labels indicating whether the AG dimer at index 61 is a splice site or not.

the AG dimer at sixty-first position and a label associated with the sequence that indicates whether this dimer is an acceptor splice site or not, as shown in Figure 5.4. The five organisms are *C.elegans* as source organism, and four target organisms at increasing evolutionary distance from *C.elegans*: *C.remanei*, *P.pacificus*, *D.melanogaster*, and *A.thaliana*. For the source organism there is one fold of 100,000 instances, whereas for the target organisms there are three folds with 1,000, 2,500, 6,500, 16,000, 25,000, 40,000, and 100,000 instances that can be used for training, and three corresponding folds of 20,000 instances that can be used for testing. For the target organisms the datasets with 2,500, 6,500, 16,000, 40,000, and 100,000 instances were used, as shown in Figure 5.5. In each file there are about 1% positive instances (i.e., DNA sequences in which the AG dimer at sixty-first position is an acceptor splice site) and the remaining instances are negative. Note that although the dataset used has only acceptor splice sites, the problem of predicting donor splice sites can be addressed with the same approach.

5.1.2 Experimental Setup

Two types of features were used to represent the data. In one representation, a sliding window approach was used to count the k-mer frequencies, and represent each sequence as the count of k-mer occurrences. This representation was used with this domain adaptation classifier for both problems: protein localization, and splice site prediction. The other type of features – in which each sequence was converted into a set of features that represent the

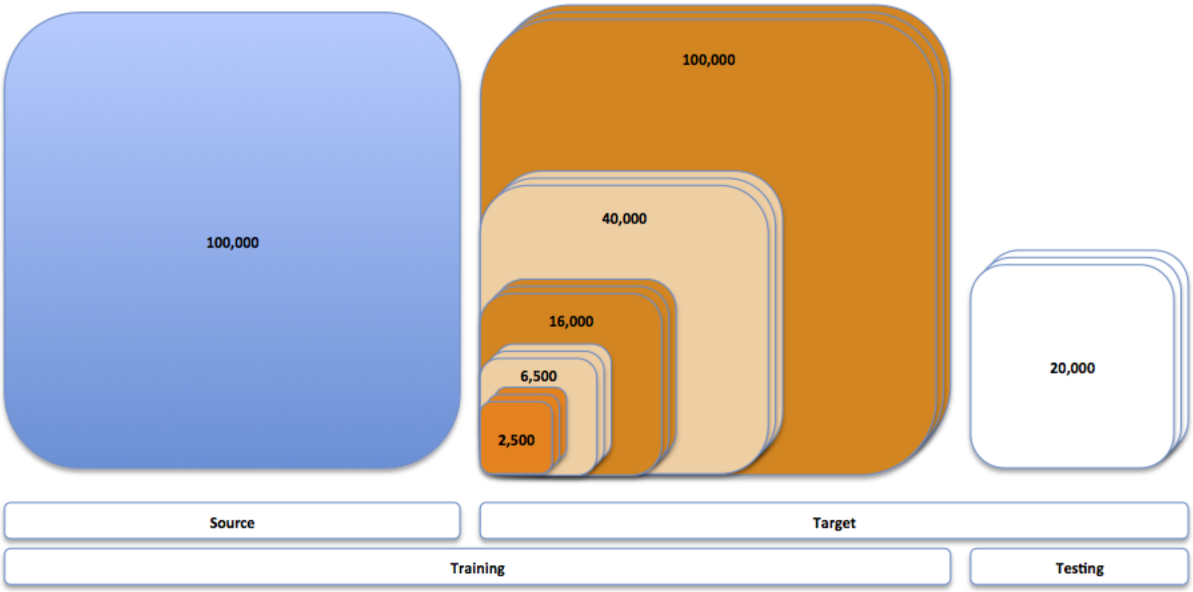


Figure 5.5: For the source organism, *C.elegans*, there is one fold of 100,000 instances, whereas for the target organisms – *C.remanei*, *P.pacificus*, *D.melanogaster*, and *A.thaliana* – there are three folds with 2,500, 6,500, 16,000, 40,000, and 100,000 instances that can be used for training, and three corresponding folds of 20,000 instances that can be used for testing. In each file there are about 1% positive instances and about 99% negative instances.

nucleotides present in the sequence at each position, and the 3-mer at each position – was used only with the other domain adaptation proposed, and for the splice site prediction.

Protein Localization

Each amino-acid sequence was represented as a count of occurrences of *k*-mers. A sliding window approach was used to count the *k*-mer frequencies. For example, a cytoplasm protein starting with `LLRSYRS`... would be represented in WEKA ARFF format, when using 2-mers, as:

```
@RELATION rel

@ATTRIBUTE 2-MER_AA

⋮

@ATTRIBUTE 2-MER_VV

@ATTRIBUTE cls {extracellular_space,cytoplasm,inner_membrane}
```

```
@DATA
1,...,1,...,2,...,1,...,1,...,cytoplasm
```

which are the counts corresponding to the occurrences of features LL, LR, RS, SY, and YR, respectively.

Splice Site Prediction

Three representations for nucleotide sequences were used. In one representation, a sliding window approach was used to count the 8-mer frequencies, and represent each sequence as the count of 8-mer occurrences (the other two representations are described in Section 5.2). For example, a DNA sequence starting with AAGATTCGC... and label -1 would be represented in WEKA ARFF format as:

```
@RELATION rel
@ATTRIBUTE 8-MER_AAAAAAAA
⋮
@ATTRIBUTE 8-MER_AAGATTCG
⋮
@ATTRIBUTE 8-MER_AGATTCGC
⋮
@ATTRIBUTE 8-MER_TTTTTTTT
@ATTRIBUTE cls {1,-1}
@DATA
1,...,1,...,-1
```

which are the counts corresponding to the occurrences of 8-mers AAGATTCG and AGATTCGC, respectively. This representation was used with only this semi-supervised domain adaptation classifier.

Data Splits and Metrics Used

Protein Localization

In order to obtain unbiased estimates for classifier performance five-fold cross validation was used. All labeled data from the source domain for training (tSL) was used and the target domain data was randomly split into 3 sets: up to 20% used as labeled data for training (tTL), up to 60% used as unlabeled data for training (tTU), and 20% used as test data (TT), and the classifier was trained on tSL + tTL + tTU and tested on TT, as shown in Figure 5.6.

To evaluate the classifier with these data, the area under the receiver operating characteristic (auROC) was used, as the class distributions are relatively balanced.

Splice Site Prediction

For the splice site prediction there are 3 folds for each target organism. To limit the number of experiments, the following datasets were used:

- The 100,000 *C.elegans* instances as source labeled data used for training (tSL).
- Only the sets with 2,500, 6,500, 16,000, and 40,000 instances as labeled target data used for training (tTL).

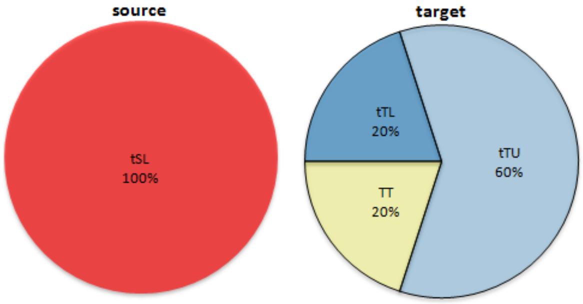
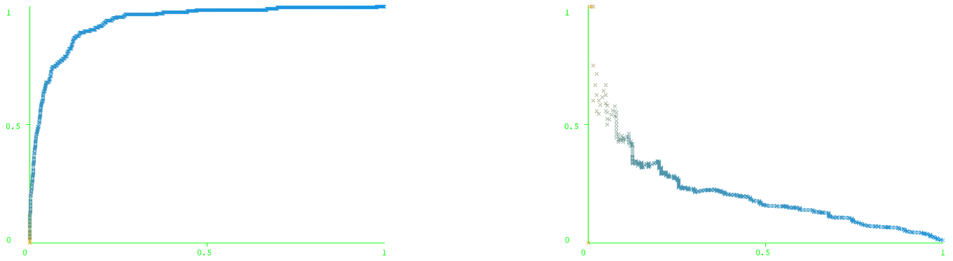


Figure 5.6: Experimental setup for protein localization: 3 datasets are used to train the classifier – source domain labeled (tSL), target domain labeled (tTL), and target domain unlabeled (tTU) – and 1 to test it – target domain labeled (TT).



(a) First fold: auROC = 0.9331 (b) First fold: auPRC = 0.2132

Figure 5.7: auROC and auPRC values for supervised domain adaptation classifier derived from the regularized logistic regression, described in Section 5.3.2. This classifier was trained with one of the three folds of target labeled data from *A.thaliana* with 2,500 instances. The auROC is 0.9331, suggesting a highly accurate classifier. A more accurate picture, in terms of classifier's performance for imbalanced datasets, is given by auPRC. Its corresponding value is 0.2132.

- The set with 100,000 instances as unlabeled target data used for training (tTU), and also as validation dataset.
- The corresponding folds of the 20,000 instances as target data used for testing (TT).

Then the results are averaged over the 3 folds to obtain unbiased estimates. For semi-supervised algorithms, the classifier was trained on tSL + tTL + tTU and tested on TT, whereas for supervised algorithms, the classifier was trained on tSL + tTL and tested on TT.

To evaluate the classifiers with these data, the area under precision-recall curve (auPRC) was used, a metric that is preferred over area under a receiver operating characteristic curve when the class distribution is skewed (Davis and Goadrich 2006), as shown in Figure 5.7.

Research Questions

This experimental setup was used to answer several general questions (Q1, Q2, and Q3), and several classifier specific questions (Q4, Q5, and Q6). Specifically, how does the performance of the classifier vary with:

Q1 Features used?

Q2 Variation with the amount of target labeled/unlabeled data?

Q3 The distance between the source and target domains?

Q4 Number of features used in the target domain (i.e., keep all features, remove at most 50% of the least occurring features)?

Q5 Number of features retained in the source domain after selecting the generalizable features?

Q6 The choice of the source and target domains (for protein localization)?

Protein Localization

As baselines, this classifier was compared with the multinomial naïve Bayes classifier trained on all source data, NB_{tSL}, the multinomial naïve Bayes classifier trained on 5% target data, $\text{NB}_{tTL(5\%)}$, and the multinomial naïve Bayes classifier trained on 80% target data, $\text{NB}_{tTL(80\%)}$. Each classifier was tested on 20% of target data. The expectation is that the prediction accuracy of this classifier will be lower bounded by $\text{NB}_{tTL(5\%)}$, upper bounded by $\text{NB}_{tTL(80\%)}$, and better than NB_{tSL}.

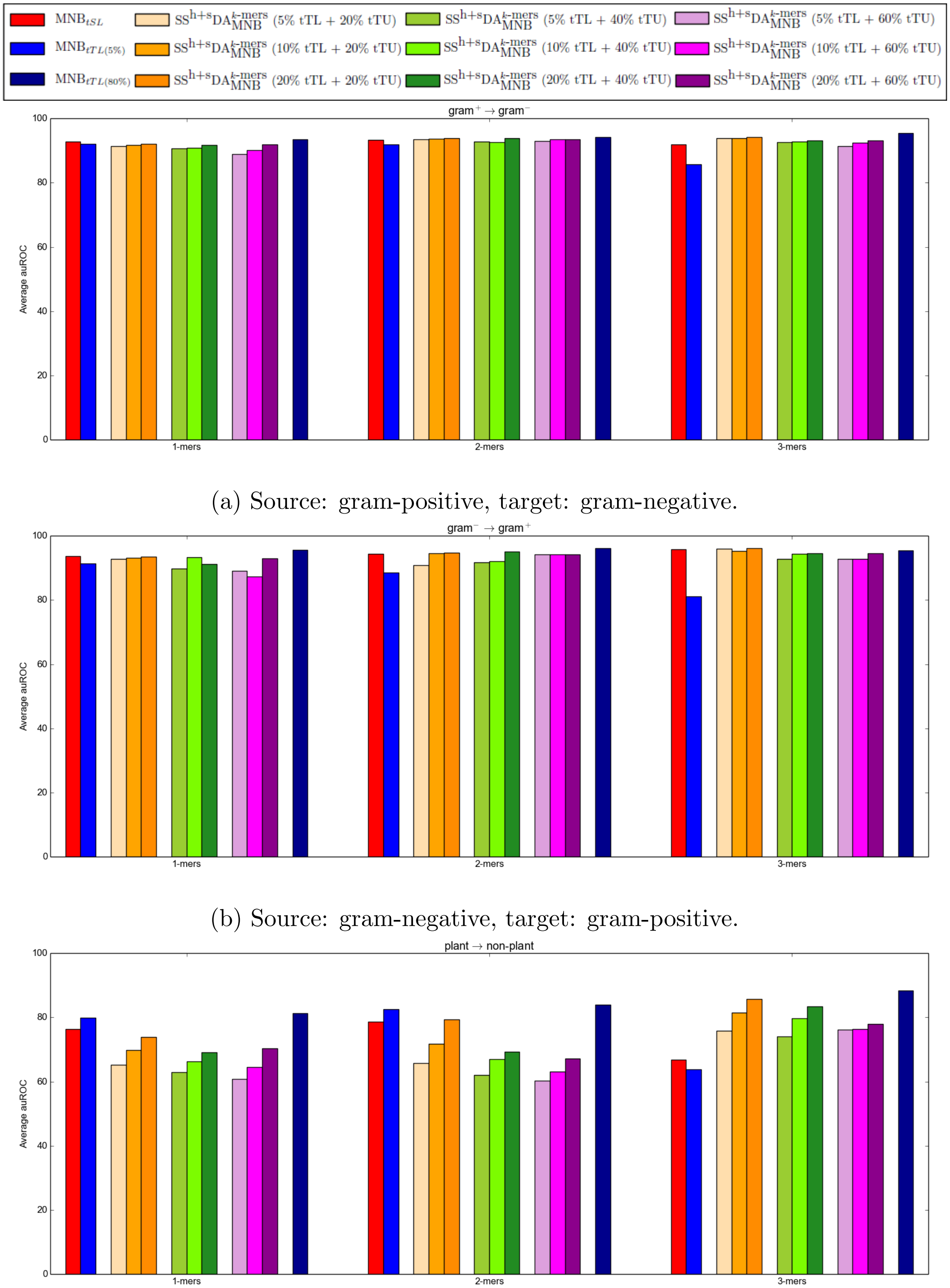
Splice Site Prediction

As baseline, this classifier was compared with the best overall algorithm in Schweikert et al. (2008).

5.1.3 Results and Discussion

Protein Localization

Table 5.1 and Figure 5.8 show the average auROC values over the five-fold cross validation trials for this algorithm and for the baseline algorithms. From these results the following observations can be made:



(a) Source: gram-positive, target: gram-negative.

(b) Source: gram-negative, target: gram-positive.

(c) Source: plant, target: non-plant.

Figure 5.8: Results of the semi-supervised domain adaptation classifier derived from multinomial naïve Bayes with counts of k-mer features on protein data.

Table 5.1: A comparison, on the protein localization task, between the domain adaptation classifier, $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$, the multinomial naïve Bayes classifier trained on all source data, MNB_{tSL}, the multinomial naïve Bayes classifier trained on 5% target data $\mathrm{MNB}_{tTL(5\%)}$, and the multinomial naïve Bayes classifier trained on 80% target data, $\mathrm{MNB}_{tTL(80\%)}$. The results are reported as average auROC values over five-fold cross validation trials. The best values for the domain adaptation classifier are highlighted. Note that $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$ is bounded by $\mathrm{MNB}_{tTL(5\%)}$ and $\mathrm{MNB}_{tTL(80\%)}$, and that $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$ predicts more accurately as the length of k-mers increases.

(a) PSORTb dataset: source domain is gram^{+} and target domain is gram^{-}

Features	Classifier	Unlabeled	Labeled		
			5%	10%	20%
count of 1-mers	MNB_{tSL}		92.74		
	$\mathrm{MNB}_{tTL(5\%)}$		92.18		
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$	20%	91.42	91.70	**92.08**
		40%	90.68	90.82	91.68
		60%	89.00	90.20	91.90
	$\mathrm{MNB}_{tTL(80\%)}$		93.52		
count of 2-mers	MNB_{tSL}		93.30		
	$\mathrm{MNB}_{tTL(5\%)}$		91.90		
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$	20%	93.58	93.66	**93.94**
		40%	92.84	92.68	93.90
		60%	92.92	93.58	93.50
	$\mathrm{MNB}_{tTL(80\%)}$		94.24		
count of 3-mers	MNB_{tSL}		91.94		
	$\mathrm{MNB}_{tTL(5\%)}$		85.80		
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{k\text{-mers}}_{\mathrm{MNB}}$	20%	93.80	93.80	**94.24**
		40%	92.62	92.78	93.14
		60%	91.34	92.40	93.08
	$\mathrm{MNB}_{tTL(80\%)}$		95.52		

A1 In terms of features used, the best results were obtained when using 3-mers as features – an example is shown in Figure 5.9. This makes sense since longer k-mers capture more information associated with the relative position of each amino-acid. When using 3-mers, the proposed algorithm provides between 9.84% and 34.14% better classification accuracy when compared to multinomial naïve Bayes classifier trained on 5% of the labeled data from the target domain, and between 0.37% and 28.2% when compared to the multinomial naïve Bayes classifier trained on labeled data from the

Table 5.1: (Cont.)

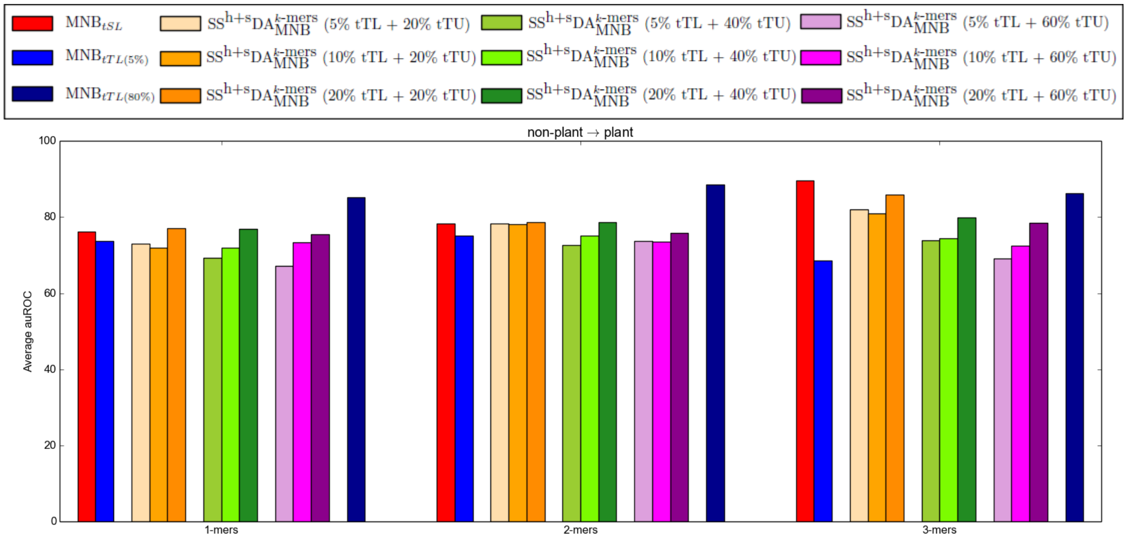
(b) PSORTb dataset: source domain is gram$^-$ and target domain is gram$^+$

Features	Classifier	Unlabeled	Labeled		
			5%	10%	20%
count of 1-mers	MNB$_{tSL}$		93.60		
	MNB$_{tTL(5\%)}$		91.42		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	92.78	93.20	**93.46**
		40%	89.78	93.26	91.18
		60%	89.12	87.28	93.02
	MNB$_{tTL(80\%)}$		95.56		
count of 2-mers	MNB$_{tSL}$		94.42		
	MNB$_{tTL(5\%)}$		88.52		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	90.90	94.52	94.66
		40%	91.80	92.06	**95.02**
		60%	94.26	94.28	94.28
	MNB$_{tTL(80\%)}$		96.16		
count of 3-mers	MNB$_{tSL}$		95.78		
	MNB$_{tTL(5\%)}$		81.18		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	95.90	95.20	**96.14**
		40%	92.80	94.40	94.60
		60%	92.78	92.82	94.60
	MNB$_{tTL(80\%)}$		95.44		

source domain, except when the plant proteins are the target domain.

A2 For most cases, the largest auROC values for this algorithm were obtained when using the least amount of target unlabeled data – an example is shown in Figure 5.10a. This would suggest that even though using unlabeled data is beneficial, using too much unlabeled data is detrimental because the unlabeled instances may act as noise and corrupt the prediction from the target labeled data. In addition, intuitively, using more labeled data from the target domain should lead to better prediction accuracy. This was indeed the case with this classifier – as shown in the example in Figure 5.10b.

A3 When the source and target domains are close the classifier learned is better. For example, the auROC is higher for the PSORTb datasets than for the TargetP datasets.



(d) Source: non-plant, target: plant.

Figure 5.8: (Cont.)

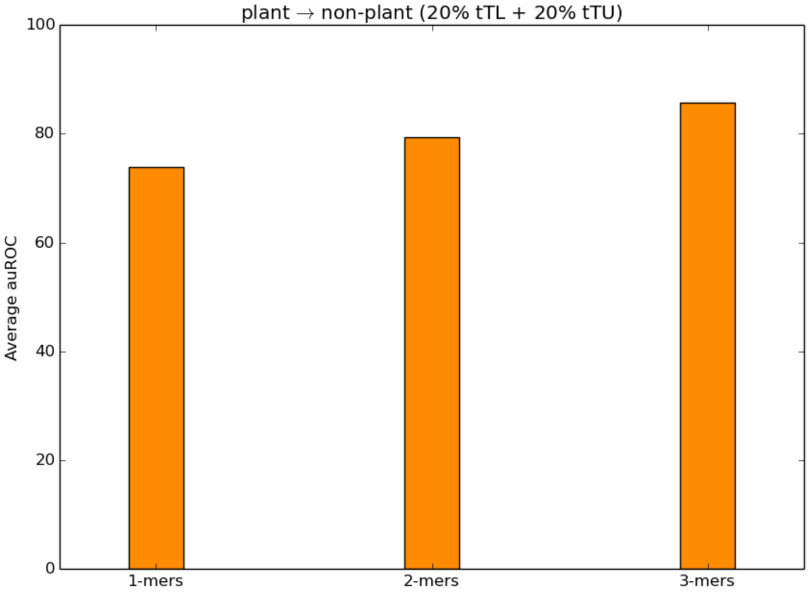
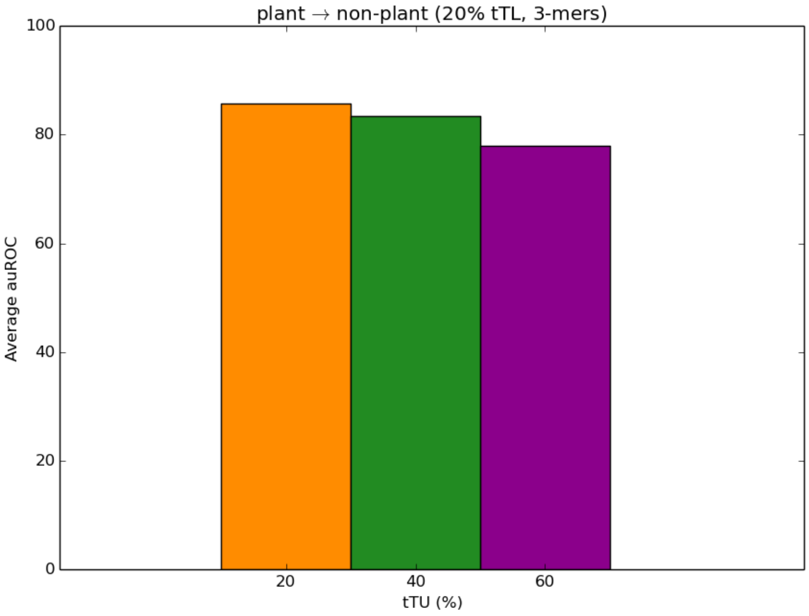
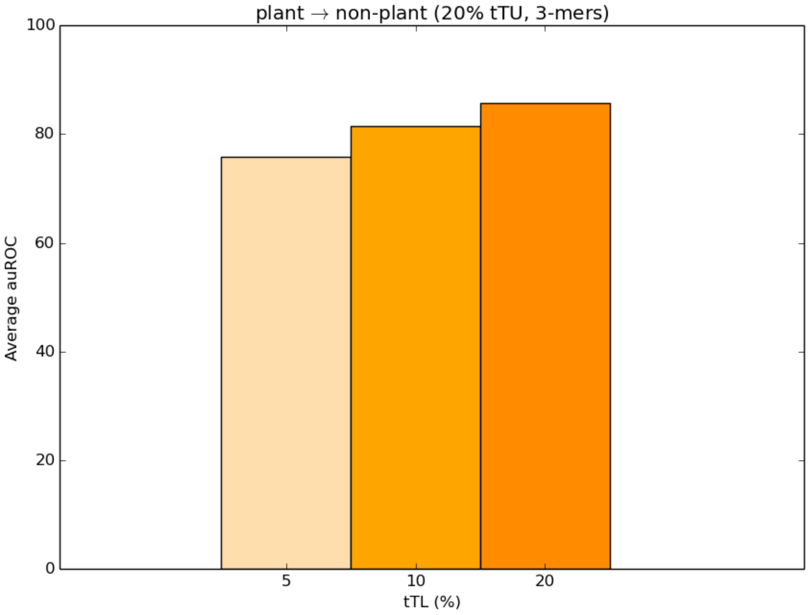


Figure 5.9: Comparison between results obtained when using 1-mers, 2-mers, and 3-mers as features with the semi-supervised domain adaptation classifier derived from multinomial naïve Bayes. Best results were obtained using 3-mers.



(a) Best results were obtained when using the least amount of target unlabeled data.



(b) Best results were obtained when using the most amount of target labeled data.

Figure 5.10: Comparison between results obtained when using different amounts of target labeled, and target unlabeled data with the semi-supervised domain adaptation classifier derived from multinomial naïve Bayes.

Table 5.1: (Cont.)

(c) TargetP dataset: source domain is plant and target domain is non-plant

Features	Classifier	Unlabeled	Labeled		
			5%	10%	20%
count of 1-mers	MNB$_{tSL}$		76.38		
	MNB$_{tTL(5\%)}$		79.90		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	65.26	69.84	**73.98**
		40%	62.90	66.24	69.16
		60%	60.88	64.52	70.40
	MNB$_{tTL(80\%)}$		81.28		
count of 2-mers	MNB$_{tSL}$		78.62		
	MNB$_{tTL(5\%)}$		82.60		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	65.78	71.84	**79.38**
		40%	62.12	67.02	69.34
		60%	60.28	63.08	67.14
	MNB$_{tTL(80\%)}$		83.96		
count of 3-mers	MNB$_{tSL}$		66.82		
	MNB$_{tTL(5\%)}$		63.86		
	SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$	20%	75.82	81.44	**85.66**
		40%	74.04	79.72	83.46
		60%	76.18	76.36	77.96
	MNB$_{tTL(80\%)}$		88.36		

Therefore, the closer the target domain is to the source domain the better the classifier learned.

A4 When trying to establish how many features from the target domain should be used it was determined that removing any features does not improve the performance of the proposed algorithm.

A5 When trying to ascertain how many features from the source domain should be kept after ranking them with Equation (5.1), it was determined that the best results were obtained when at least 50% of the features were kept, i.e., the 50% top-ranked features and any other features with the same rank as the last feature kept.

A6 For the PSORTb dataset, the SS$^{\text{h+s}}$DA$^{k\text{-mers}}_{\text{MNB}}$ classifier had better prediction accuracy

Table 5.1: (Cont.)

(d) TargetP dataset: source domain is non-plant and target domain is plant

Features	Classifier	Unlabeled	Labeled		
			5%	10%	20%
count of 1-mers	MNB_{tSL}		76.18		
	$\text{MNB}_{tTL(5\%)}$		73.66		
	$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	20%	72.96	71.90	**77.04**
		40%	69.22	71.96	76.96
		60%	67.16	73.40	75.48
	$\text{MNB}_{tTL(80\%)}$		85.14		
count of 2-mers	MNB_{tSL}		78.36		
	$\text{MNB}_{tTL(5\%)}$		75.08		
	$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	20%	78.24	78.10	**78.68**
		40%	72.72	75.14	78.62
		60%	73.80	73.62	75.92
	$\text{MNB}_{tTL(80\%)}$		88.52		
count of 3-mers	MNB_{tSL}		89.68		
	$\text{MNB}_{tTL(5\%)}$		68.60		
	$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	20%	82.00	80.92	**85.96**
		40%	73.82	74.42	79.90
		60%	69.04	72.56	78.48
	$\text{MNB}_{tTL(80\%)}$		86.28		

when the gram-negative proteins were used as the source domain than when the gram-positive proteins were used as the source domain. Similarly, for the TargetP dataset, better predictions were obtained when using non-plant proteins as the source domain than when using plant proteins as the source domain. This is because in both cases there were more gram-negative instances and more non-plant instances, respectively, than gram-positive instances and plant instances, respectively.

It is interesting to note that in some instances the multinomial naïve Bayes classifier trained on the source domain performed better than this algorithm. This occurred mainly when this algorithm used 5% or 10% of the target labeled data and when the features were 1-mers or 2-mers. However, this is somewhat expected, as using very little labeled data from the target domain does not provide a representative sample for the population, and using short

k-mers does not capture the relative position of the amino-acids.

Splice Site Prediction

Although this algorithm worked well for the protein localization task, it performed poorly on the splice site prediction task, as shown in Table 5.2. This algorithm always gravitated towards classifying each instance as not containing a splice site. This is due mainly because the 8-mers indicating a splice site occur with low frequency and their relative position to the splice site is important.

From these results the following observations can be made:

A2 The auPRC values for this algorithm were very similar regardless of the amount of target labeled data.

A3 The classification performance of this algorithm did not decrease as the distance between the source and target domains increased, as one would have expected.

A4 Similar to protein localization task, removing any features does not improve the performance of this algorithm.

A5 In terms of the number of features from the source domain to keep after ranking them with Equation (5.1), it was determined that the best results were obtained when at least 50% of the features were kept, i.e., the 50% top-ranked features and any other features with the same rank as the last feature kept.

The last two observations suggest that the features need to take into consideration the locations of the 8-mers to improve the classification accuracy of this classifier on the splice site prediction task.

Table 5.2: auPRC values for the 4 target organisms based on the number of labeled target instances used for training: 2,500, 6,500, 16,000, and 40,000. For comparison with this algorithm the values for the best overall supervised domain adaptation algorithm in Schweikert et al. (2008), $\text{SVM}_{\text{S,T}}$, are shown as S_DA_{SVM}.

(a) *C.remanei*

	2,500	6,500	16,000	40,000
$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	1.13	1.13	1.13	1.10
S_DA_{SVM}	77.06	77.80	77.89	79.02

(b) *P.pacificus*

	2,500	6,500	16,000	40,000
$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	1.00	0.97	1.07	1.10
S_DA_{SVM}	64.72	66.39	68.44	71.00

(c) *D.melanogaster*

	2,500	6,500	16,000	40,000
$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	1.07	1.13	1.07	1.03
S_DA_{SVM}	40.80	37.87	52.33	58.17

(d) *A.thaliana*

	2,500	6,500	16,000	40,000
$\text{SS}^{\text{h+s}}\text{DA}^{k\text{-mers}}_{\text{MNB}}$	1.20	1.17	1.20	1.17
S_DA_{SVM}	24.21	27.30	38.49	49.75

5.2 Semi-supervised Domain Adaptation Classifiers Derived from Naïve Bayes with Location-Aware Features for the task of Splice Site Prediction

One major drawback of the previous algorithm is that during the first iterations it assigns low weight to the target data, including the labeled data, through λ in (Equations (5.3) and (5.4)). This biases the classifier towards the source domain. However, it is not effective to only assign a different weight to the target labeled data in Equations (5.3) and (5.4). This is because when there are much more labeled instances from the source domain, as well as much more unlabeled instances from the target domain, i.e., $m_{tSL} \gg m_{tTL}$ and

$m_{tTU} \gg m_{tTL}$, the sums and counts for the target labeled data in these two equations are much smaller than their counterparts for the source data and target unlabeled data, rendering the weight assignment useless. Instead, these values need to also be normalized, or better yet, use their probabilities. Thus, the prior is estimated as

$$P(y^i = y) = \beta P_{tTL}(y^i = y) + (1-\beta)[(1-\lambda)P_{tSL}(y^i = y) + \lambda P_{tTU}(y^i = y)] \tag{5.7}$$

and the likelihood as

$$\begin{aligned} P(x_j \mid y^i = y) &= \beta P_{tTL}(x_j \mid y^i = y) \\ &\quad + (1-\beta)[(1-\lambda)P_{tSL}(x_j \mid y^i = y) + \lambda P_{tTU}(x_j \mid y^i = y)] \end{aligned} \tag{5.8}$$

where $\beta \in (0,1)$ is a constant weight, and λ is defined the same as in the previous approach.

In addition to using different formulas for prior and likelihood, a second change was made to the previous algorithm. The measure for ranking the features, Equation (5.1), was replaced with the following measure in Equation (5.9). This change was made because the goal of ranking the features is to select top features in terms of their correlation with the class, or assign them different weights: higher weights to features that are more correlated with the class, and lower weights to the features that are less correlated with the class. Therefore, the mutual information (Shannon 1948) of each feature is a more appropriate measure in determining the correlation of the feature with the class rather than the marginal probability of the feature. With this new formula, the features are ranked better based on their generalizability between the source and target domains:

$$f(x_j) = \frac{I_{tSL}(x_j; y) \cdot I_{tTL}(x_j; y)}{|I_{tSL}(x_j; y) - I_{tTL}(x_j; y)| + \alpha} \tag{5.9}$$

Algorithm 2 Outline of the updated semi-supervised domain adaptation classifier derived from naïve Bayes, $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$.

```
1: Select the features to be used from the source domain using Equation (5.9).
2: Simultaneously compute the prior and likelihood, using Equations (5.7) and (5.8), respectively. Note that for the source domain all labeled instances are used but only with the features selected in step 1, whereas for the target domain only labeled instances are used with all their features.
3: Assign labels to the unlabeled instances from the target domain using Equation (5.6). Note that self-training is used, i.e., a number of instances, proportional to the class distribution, with the highest class probability are considered to be labeled in subsequent iterations (i.e., assign them hard-labels); assign soft-labels to the remaining instances.
4: while labels assigned to unlabeled data change do
5:    M-step: Same as step 2, except that the instances in the target unlabeled dataset are also used; for this dataset the class for the self-trained instances is used, and the class distribution for the rest of the instances.
6:    E-step: Same as step 3.
7: end while
8: Use classifier on new target instances.
```

where

$$I_d(x_j; y) = \sum_{y \in \mathcal{Y}} P_d(x_j, y) \log \frac{P_d(x_j, y)}{P_d(x_j) P_d(y)}$$

is the mutual information between feature x_j and class y, and $d \in \{tSL, tTL\}$. The numerator in Equation (5.9) ranks higher the features that have higher mutual information in their domains, whereas the denominator ranks higher the features that have closer mutual information between the domains.

Instead of ranking the features and using the top ranked features, an alternative is to assign different weights to the features and higher values for $f(x_j)$ computed. An updated formula for ranking the features was proposed as most of the mutual information values are close to zero since each feature contributes very little to the classification of an instance. Therefore, the nominator is about one order of magnitude smaller than the denominator in Equation (5.9), whereas in Equation (5.10), the numerator and denominator have the same order of magnitude, resulting in higher values for $f(x_j)$.

The final change made to the previous method was the use of location-aware features

Algorithm 3 Outline of the updated semi-supervised domain adaptation classifier derived from naïve Bayes with weighted features, $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.9)}}$ and $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$.

```
1: *Assign different weights to the features of the source dataset* using either Equation (5.9)
   or Equation (5.10).
2: Simultaneously compute the prior and likelihood, with Equations (5.7) and (5.8), respectively.
   Note that for the source domain all labeled instances are used *but the features are
   assigned different weights in step 1*, whereas for the target domain only labeled instances
   are used with all their features.
3: Assign labels to the unlabeled instances from the target domain using Equation (5.6).
   Note that self-training is used, i.e., a number of instances, proportional to the class
   distribution, with the highest class probability are considered to be labeled in subsequent
   iterations (i.e., assign them *hard*-labels); assign *soft*-labels to the remaining instances.
4: **while** labels assigned to unlabeled data change **do**
5:     ***M*-step**: Same as step 2, except that the instances in the target unlabeled dataset
       are also used; for this dataset the class for the self-trained instances is used, and the
       class distribution for the rest of the instances.
6:     ***E*-step**: Same as step 3.
7: **end while**
8: Use classifier on new target instances.
```

instead of counts of 8-mers. As opposed to the previous representation that leads to 4^8 or 65,536 *sparse features* that do not take into consideration the location of each nucleotide, the other two representations (used with this proposed methods and the subsequent ones) are “location-aware”, with less features. In one of them, each DNA sequence was converted into a set of features that represent the nucleotides present in the sequence at each position, and the 3-mer at each position. For example, the same DNA sequence and label `-1` would be represented as:

```
@RELATION rel

@ATTRIBUTE NUCLEOTIDE_1 {A,C,G,T}

⋮

@ATTRIBUTE NUCLEOTIDE_141 {A,C,G,T}

@ATTRIBUTE 3-MER_1 {AAA,AAC,...,TTT}

⋮

@ATTRIBUTE 3-MER_139 {AAA,AAC,...,TTT}
```

```
@ATTRIBUTE cls {1,-1}
@DATA
A,A,G,A,T,T,C,G,C,...,AAG,AGA,GAT,...,-1
```

These features create a compact representation of a balanced combination of simple features in each DNA sequence, i.e., the nucleotides, and more complex features – features that capture the correlation between the nucleotides, i.e., the 3-mers.

$$f(x_j) = \frac{I_{tSL}(x_j; y) \cdot I_{tTL}(x_j; y)}{(I_{tSL}(x_j; y) - I_{tTL}(x_j; y))^2 + \alpha} \tag{5.10}$$

With these changes, the two algorithms proposed use the source labeled data and the target labeled and unlabeled data to train a classifier for the target domain. For the source domain, Algorithm 2 selects generalizable features to be used, whereas Algorithm 3 assigns different weights to the features. These differences are highlighted in steps 1 and 2 of the algorithms by using italics text.

5.2.1 Balance Class Distribution with Ensemble Learning

Considering that the data is highly imbalanced, this method was also evaluated in an ensemble learning setting. An approach similar to Breiman (1996) was used. The labeled data was sampled with no replacement, from the source and target domains, and generated 99 balanced subsets, with 50% positive instances and 50% negative instances each. That is, in each balanced subset all positive instances from the initial set were included along with 1% negative instances sampled with no replacement from the initial set. 99 balanced subsets were generated because the ratio of positive to negative instances is 1 to 99. Then this algorithm was run 99 times with sampled labeled data and whole unlabeled and test data sets from the target domain. When predicting the class for a new instance the majority voting of the 99 algorithms was used, as shown in Figure 5.11.

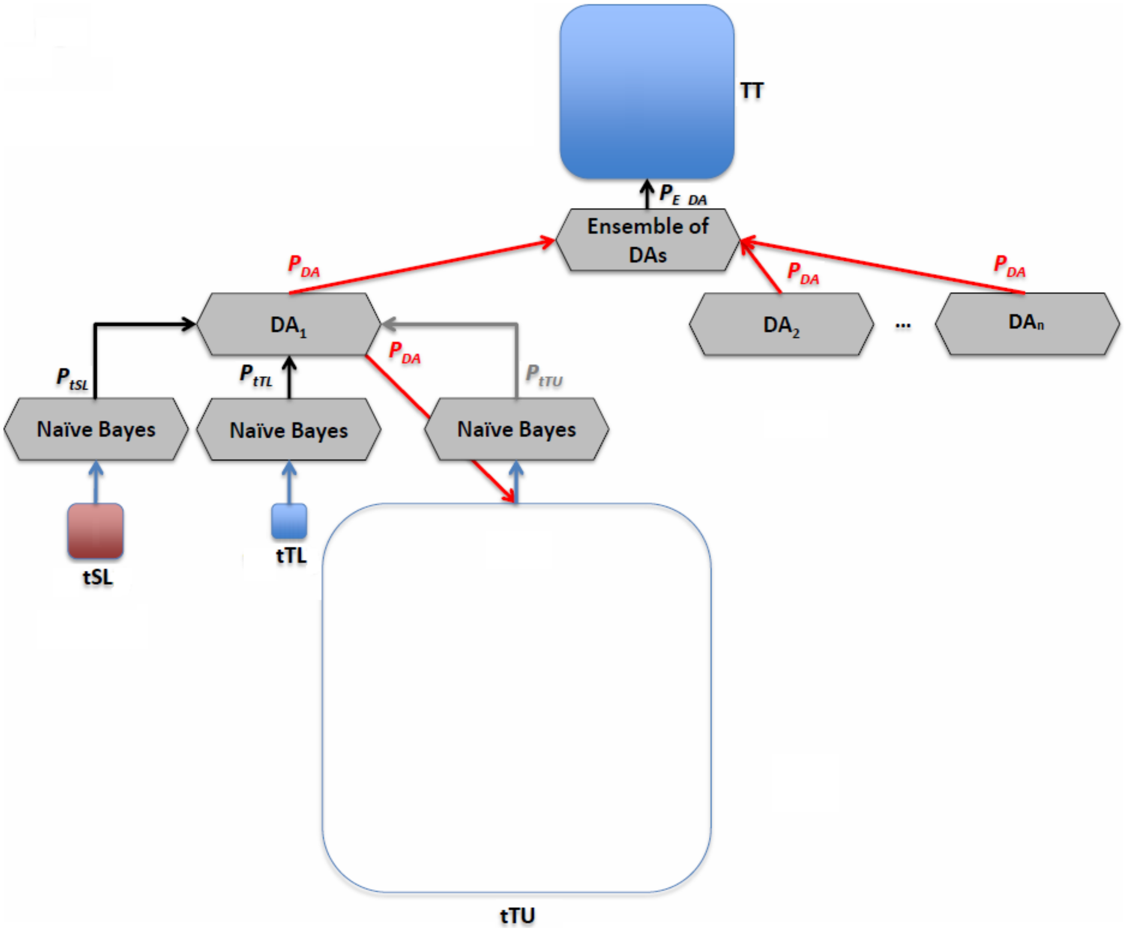


Figure 5.11: Ensemble of semi-supervised domain adaptation classifiers derived from naïve Bayes.

5.2.2 Experimental Setup

Two different experiments were run with a grid search for the optimal values for $\beta \in \{0.1, 0.2, \ldots, 0.9\}$ and $\delta \in \{0.01, 0.02, \ldots, 0.08, 0.09, 0.1, 0.2\}$. In the first one, only the nucleotides were used as features, whereas in the second one the nucleotides and 3-mers were used as features. In both settings, Algorithm 2 was run to select the features in the source domain, Algorithm 3 using Equation (5.9) to weigh the features in the source domain, and Algorithm 3 with Equation (5.10) to weigh the source domain features.

Two baselines were used to evaluate these classifiers. The first baseline, which was expected to represent the lower bound, was the naïve Bayes classifier trained on the target labeled data, NB_{tTL}. The assumption is that this will be the lower bound for these classifiers because it is expected that adding the labeled data from a related organism, the source domain, as well as unlabeled data from the same organism, the target domain, should produce a better classifier. The second baseline is the naïve Bayes classifier trained on the source labeled data, NB_{tSL}. It is expected that the more distantly related the two organisms are, the less accurate the classifier trained on the source data would be when tested on the target data.

In addition, the following two sets of experiments were also performed. In one set of experiments these algorithms were tested versus the null hypothesis, i.e., all nucleotides in each instance were shuffled except the AG dimer at index 61, and the labels were also shuffled, then tested these algorithms on these data sets. The intuition is that in this case these algorithms should not learn any meaningful patterns, and thus obtain poor results when evaluated on the test data. In another set of experiments the best-overall algorithm was evaluated on how it performed in an ensemble setting with balanced data. For this, the source and target labeled data were split into 99 class-balanced data sets, and used an ensemble of Algorithm 3 classifiers to predict the labels for the test data. These experiments were used to investigate if this approach is better suited for highly unbalanced data sets.

The experimental setup described below was used to answer several questions specific to this classifier, in addition to the general questions listed in Section 5.1.2 – namely, how does the performance of the classifier vary with:

Q7 Using source labeled data and target unlabeled data when training the classifier.

Q8 Using real data versus shuffled data with shuffled labels.

Q9 Using unbalanced data sets or creating balanced subsets and using an ensemble of classifiers.

Q10 Algorithm used, i.e., Algorithm 2 which keeps only the generalizable features in the source domain, or Algorithm 3 which assigns different weights to the features in the source domain.

5.2.3 Results and Discussion

Table 5.3 and Figure 5.12 list the auPRC for these classifiers, for the best overall algorithm in Schweikert et al. (2008), $\mathrm{SVM}_{\mathrm{S,T}}$, and for the classifier in Section 5.1.

Table 5.3: auPRC values for four target organisms based on the number of labeled target instances used for training: 2,500, 6,500, 16,000, and 40,000. The highest values for the domain adaptation algorithms are displayed in bold font. NB_{tTL} and NB_{tSL} are baseline naïve Bayes classifiers trained on target labeled and source data, respectively, for the two algorithms proposed. For comparison with these algorithms, the values for the best overall algorithm in Schweikert et al. (2008), $\mathrm{SVM}_{\mathrm{S,T}}$, and the values from the previous classifier, $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{MNB}}^{k\text{-mers}}$, are also shown. $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered+shuffled}}$ and $\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$ are the results when the data and the labels were shuffled. $\mathrm{E}^{\mathrm{bal}}\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$ are the results of the ensemble of classifiers.

(a) *C.remanei*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered+shuffled}}$	0.90	0.94	0.90	0.92
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.90	0.94	0.90	0.93
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.13	1.13	1.13	1.10
	NB_{tTL}	23.42	45.44	54.57	59.68
	$\mathrm{E}^{\mathrm{bal}}\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	**60.24**	62.27	63.36	63.83
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	59.18	63.10	63.95	63.80
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.9)}}$	35.03	46.08	54.89	59.73
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	48.92	60.83	63.06	63.59
	NB_{tSL}	63.77			
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	77.06	77.80	77.89	79.02
1- + 3-mers	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered+shuffled}}$	0.91	0.93	0.90	0.93
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.91	0.93	0.91	0.93
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.13	1.13	1.13	1.10
	NB_{tTL}	22.94	58.39	68.40	75.75
	$\mathrm{E}^{\mathrm{bal}}\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	25.83	26.08	25.99	26.09
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	45.29	**72.00**	74.83	77.07
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.9)}}$	24.96	61.45	69.11	75.91
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	49.22	70.23	**75.43**	**78.01**
	NB_{tSL}	77.67			

Table 5.3: (Cont.)

(b) *P.pacificus*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:filtered+shuffled}}$	0.91	0.92	0.96	0.91
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.90	0.91	0.95	0.92
	$\mathrm{SS^{h+s}DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.00	0.97	1.07	1.10
	NB_{tTL}	19.22	37.33	45.33	52.84
	$\mathrm{E^{bal}SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	**50.60**	50.64	52.34	55.16
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	45.32	49.82	52.09	54.62
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.9)}}$	19.85	37.51	45.64	52.91
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	37.20	48.71	52.31	55.62
	NB_{tSL}	49.12			
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	64.72	66.39	68.44	71.00
1- + 3-mers	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:filtered+shuffled}}$	0.92	0.89	0.95	0.91
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.92	0.89	0.94	0.91
	$\mathrm{SS^{h+s}DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.00	0.97	1.07	1.10
	NB_{tTL}	26.39	48.54	59.29	68.78
	$\mathrm{E^{bal}SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	23.48	23.61	23.38	23.72
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	20.21	53.29	62.33	69.88
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.9)}}$	20.16	43.95	57.44	65.80
	$\mathrm{SS^{h+s}DA}_{\mathrm{NB}}^{\mathrm{loc:weighted(5.10)}}$	20.19	**57.21**	**65.99**	**70.94**
	NB_{tSL}	67.10			

Table 5.3: (Cont.)

(c) *D.melanogaster*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered+shuffled}}$	0.89	0.93	0.96	0.91
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.89	0.91	0.96	0.90
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{MNB}}^{k\text{-mers}}$	1.07	1.13	1.07	1.03
	NB_{tTL}	14.90	26.05	35.21	39.42
	$\text{E}^{\text{bal}}\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$	33.57	36.87	39.35	39.62
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered}}$	33.31	36.43	40.32	42.37
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.9)}}$	16.27	26.21	35.12	39.16
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$	22.86	32.92	36.95	37.55
	NB_{tSL}	31.23			
	S_DA_{SVM}	40.80	37.87	52.33	58.17
1- + 3-mers	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered+shuffled}}$	0.89	0.88	0.94	0.92
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.89	0.89	0.94	0.89
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{MNB}}^{k\text{-mers}}$	1.07	1.13	1.07	1.03
	NB_{tTL}	13.87	25.00	35.28	45.85
	$\text{E}^{\text{bal}}\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$	**42.12**	**42.71**	**45.85**	47.42
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered}}$	25.83	32.58	39.10	**47.49**
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.9)}}$	15.03	26.45	34.73	42.90
	$\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$	22.53	29.47	36.18	42.92
	NB_{tSL}	34.09			

Table 5.3: (Cont.)

(d) *A.thaliana*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:filtered+shuffled}}$	0.97	0.91	0.96	0.91
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.92	0.90	0.96	0.91
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.20	1.17	1.20	1.17
	NB_{tTL}	7.20	17.90	28.10	34.82
	$\mathrm{E}^{\mathrm{bal}}\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.10)}}$	28.41	30.98	33.50	36.70
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:filtered}}$	18.46	25.04	31.47	36.95
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.9)}}$	8.42	18.39	28.22	34.79
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.10)}}$	13.61	22.28	29.05	34.66
	NB_{tSL}	11.97			
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	24.21	27.30	38.49	49.75
1- + 3-mers	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:filtered+shuffled}}$	0.91	0.90	0.97	0.90
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{shuffled labels + loc:weighted(5.10)}}$	0.91	0.90	0.96	0.90
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{MNB}}^{k\text{-mers}}$	1.20	1.17	1.20	1.17
	NB_{tTL}	3.10	8.76	28.11	40.92
	$\mathrm{E}^{\mathrm{bal}}\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.10)}}$	**39.80**	**42.27**	**43.10**	**45.05**
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:filtered}}$	3.99	13.96	33.62	43.20
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.9)}}$	2.65	8.72	29.39	40.35
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\text{loc:weighted(5.10)}}$	3.64	10.00	30.85	40.40
	NB_{tSL}	13.98			

Based on these results, the following observations were made:

A1 Using a classifier with both the nucleotides and the 3-mers as features performs better as the amount of the target labeled data increases, whereas a classifier using only

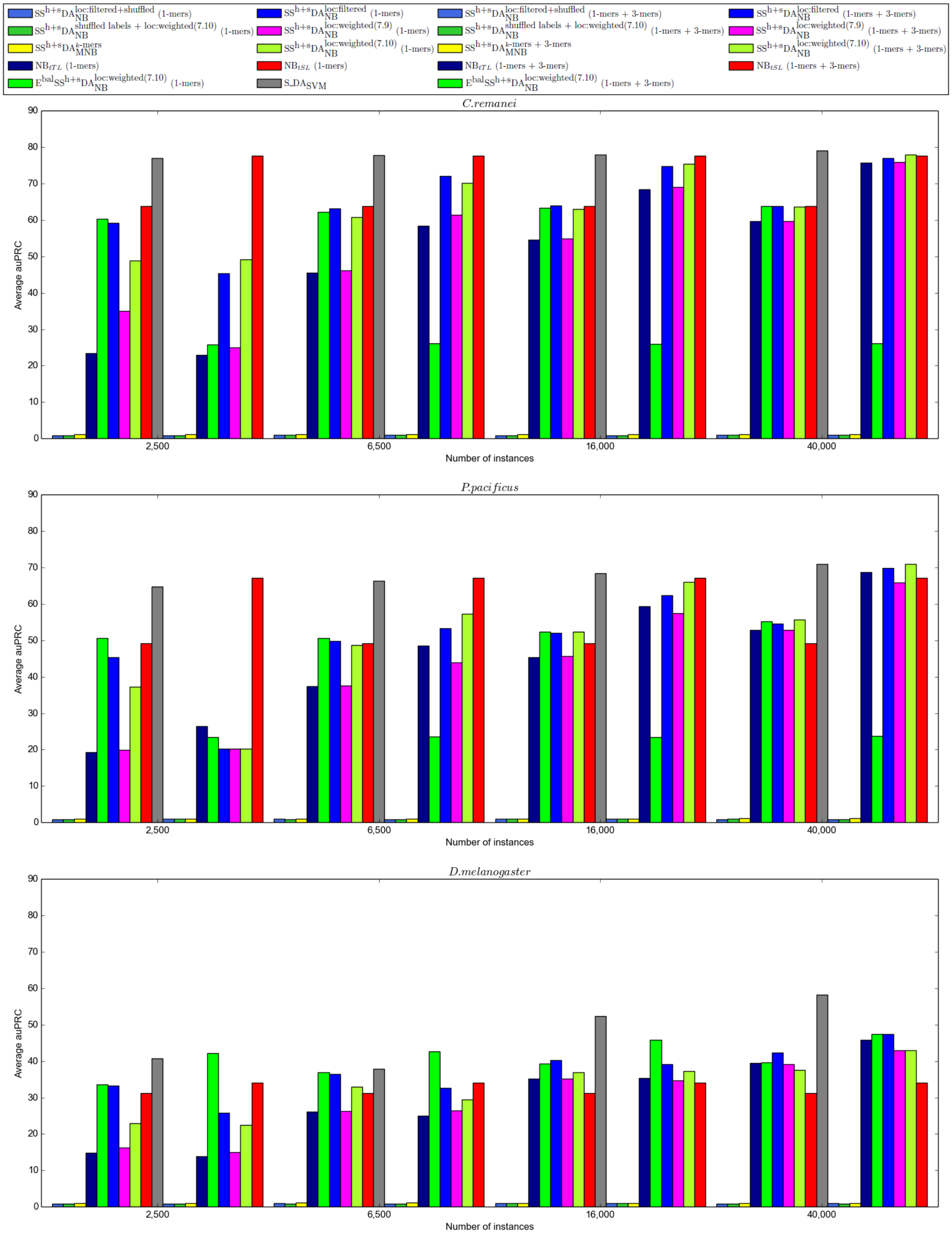


Figure 5.12: Results of the semi-supervised domain adaptation classifier derived from naïve Bayes with location-aware features on splice site data.

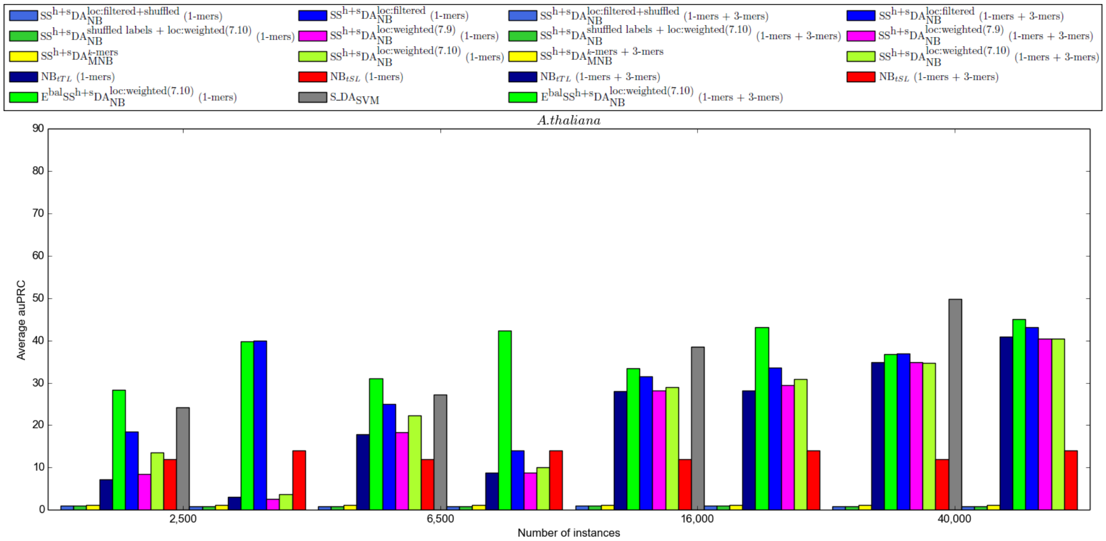


Figure 5.12: (Cont.)

the nucleotides as features performs better with smaller target labeled datasets – an example is shown in Figure 5.13. This is due to the fact that the 3-mer features are sparser than the nucleotide features, and when there is little target labeled data the classifier does not have enough data to separate the positive from the negative instances.

A2 The general trend in all classifiers is that they perform better as more target labeled instances are used for training – an example is shown in Figure 5.14. This conforms with the intuition that the more target labeled data is used the more accurate the classifier is.

A3 For the non-ensemble classifiers, as the evolutionary distance increases between the source and target organisms, the performance of these classifiers decreases. When the source and target organisms are closely related, as is the case with *C.elegans* and *C.remanei*, the large volume of labeled source data significantly contributes to generating a better classifier, compared to training a classifier on the target data alone.

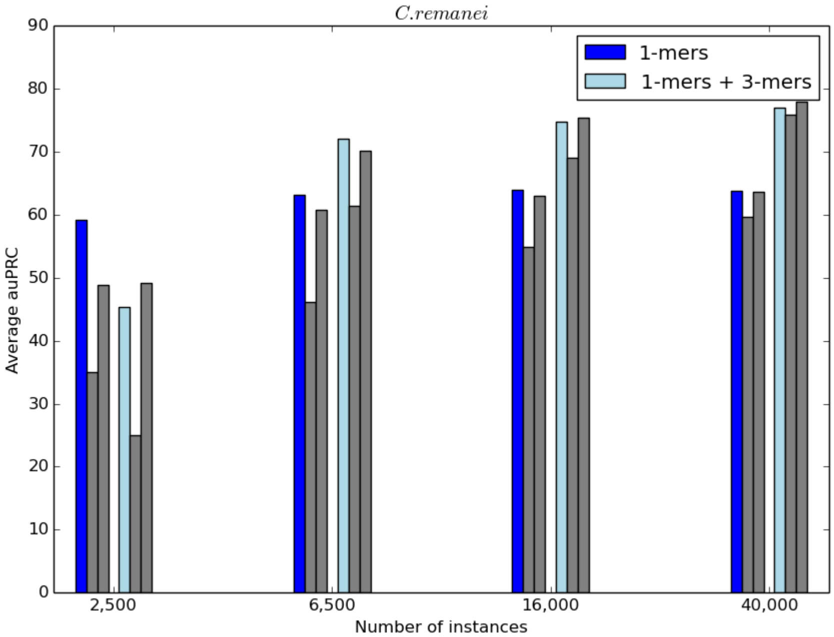


Figure 5.13: Comparison between results obtained when using 1-mers, or 1-mers and 3-mers with the semi-supervised domain adaptation classifier derived from naïve Bayes. A classifier with both the nucleotides and the 3-mers as features performs better as the amount of the target labeled data increases, whereas a classifier using only the nucleotides as features performs better with smaller target labeled datasets.

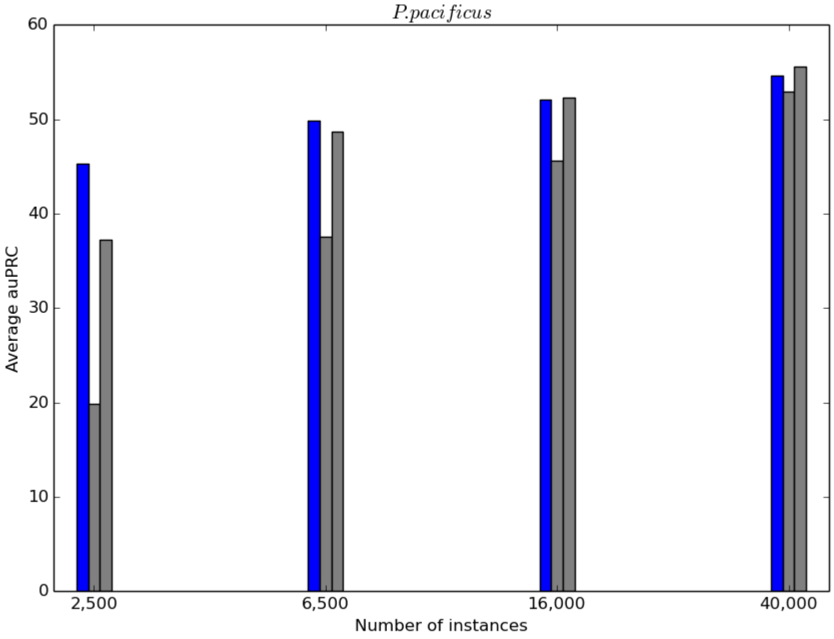


Figure 5.14: Comparison between results obtained when using different amounts of target labeled data with the semi-supervised domain adaptation classifier derived from naïve Bayes. A classifier performs better as the amount of the target labeled data increases.

However, as the source and target organisms diverge, the source data contributes less to the classifier.

A7 Using the source data and the target unlabeled data in addition to the target labeled data improves the performance of the classifier (e.g., $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered}}$ v. NB_{tTL} for *A.thaliana* with 2,500 instances) compared to training a classifier on just the target labeled data. The improvement occurs even with larger datasets, although less substantial than with smaller datasets – an example is shown in Figure 5.15.

A8 When the data and the labels were shuffled the algorithms were unable to learn meaningful patterns, resulting in poor classifiers with auPRC values less than one percent.

A9 When there are only a few labeled instances in the target domain (e.g., 2,500 instances), splitting the data into class-balanced data sets and using an ensemble of classifiers performs better than using the unbalanced data sets – an example is shown

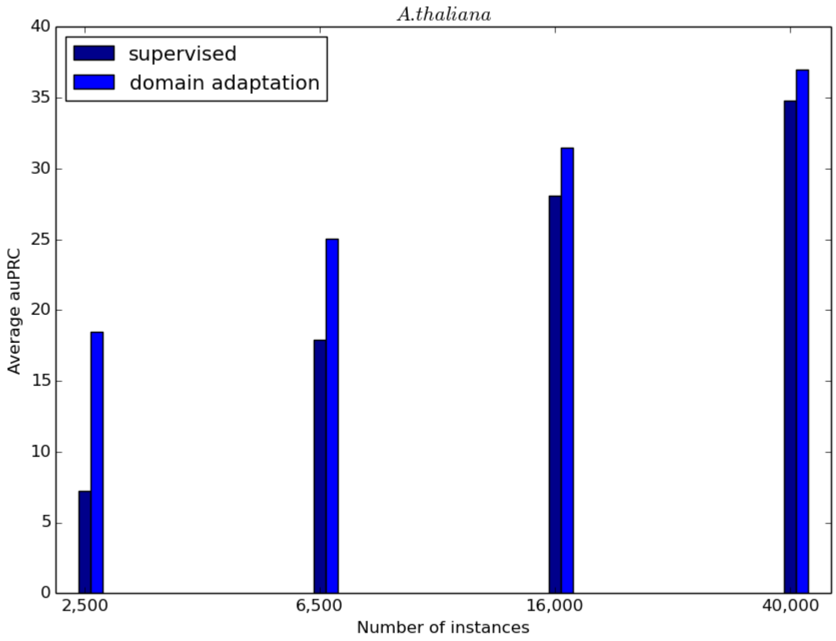


Figure 5.15: Comparison between results obtained when using the semi-supervised domain adaptation classifier derived from naïve Bayes, or the supervised naïve Bayes trained on target labeled data. Domain adaptation classifies better than supervised classifier trained on target labeled data.

in Figure 5.16. In addition, when the source and target domains are distantly related, the ensemble classifier outperforms even the S_DA$_{\text{SVM}}$ classifier (e.g., when the target domain is *D.melanogaster* and there are 2,500 or 6,500 labeled instances in the target domain, or for *A.thaliana* with 2,500 to 16,000 labeled instances in the target domain).

A10 The $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered}}$ classifier, based on Algorithm 2, and the $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$ classifier, based on Algorithm 3, performed better than the other classifier, $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.9)}}$, each producing the best results in two and five cases, respectively, when the data was not balanced – an example is shown in Figure 5.17. As mentioned above, the sparsity of the data affects the performance of the classifiers based on the features used. The $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:filtered}}$ classifier, which uses only the nucleotides as features, performs better when the amount of target labeled dataset is small, and also when the two organisms are more distantly related, whereas the $\text{SS}^{\text{h+s}}\text{DA}_{\text{NB}}^{\text{loc:weighted(5.10)}}$ classifier, which uses the

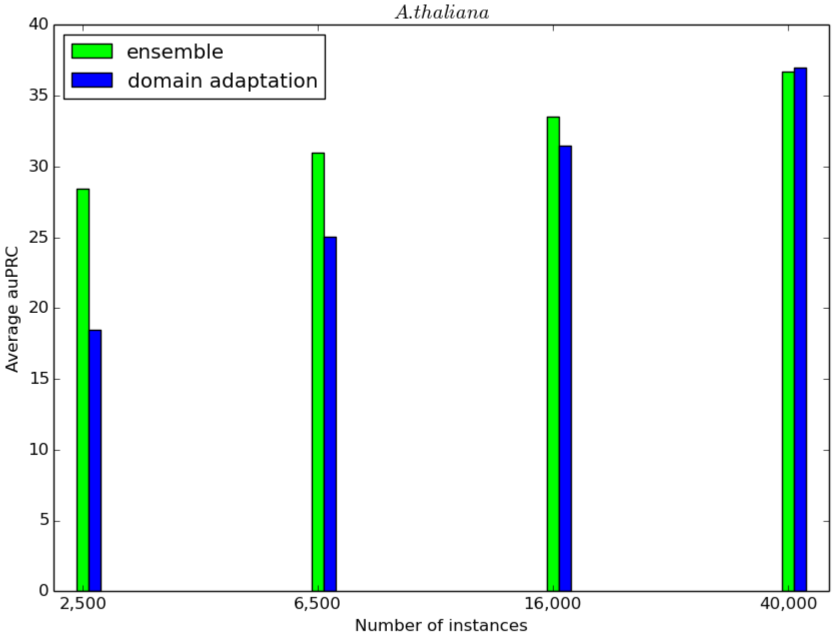


Figure 5.16: Comparison between results obtained when using the semi-supervised domain adaptation classifier derived from naïve Bayes, or the ensemble of domain adaptation classifiers. in general, ensemble classifies better than domain adaptation.

nucleotides and 3-mers as features, performs better when the target labeled dataset is larger and the two organisms are more closely related.

Although these results are not as good as the ones in Schweikert et al. (2008), they are greatly improved compared to the results in Section 5.1, yet not as much as expected. Even though they are not as good as the $\mathrm{SVM}_{\mathrm{S,T}}$ in Schweikert et al. (2008), the algorithms proposed could be superior in some contexts. For example, the complexity of SVM classifiers increases with the number of training instances and the number of features, when training the classifier (Schweikert et al. 2008). Whereas the $\mathrm{S\_DA}_{\mathrm{SVM}}$ classifier "required an equivalent of about 1,500 days of computing time on state-of-the-art CPU cores" to tune their parameters (Schweikert et al. 2008), and additional computing to analyze the biological features, the proposed algorithms required the equivalent of only 300 days of computing, and the results can be easily interpreted.

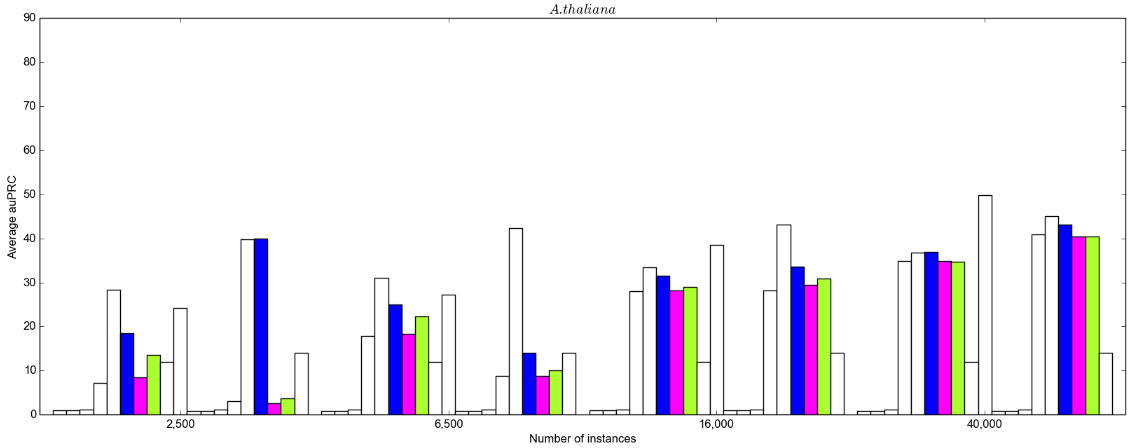


Figure 5.17: Comparison between results obtained when using the different semi-supervised domain adaptation classifiers proposed, derived from naïve Bayes. Better results were obtained with the proposed method that filtered features from source domain.

5.3 Domain Adaptation Classifiers Derived from Regularized Logistic Regression for Splice Site Prediction

After the domain adaptation algorithms derived from a generative supervised classifier (naïve Bayes), the focus was switched on domain adaptation algorithms derived from a discriminative supervised classifier (regularized logistic regression). Two such classifiers were proposed – a supervised domain adaptation classifier, described in Section 5.3.2, and a semi-supervised domain adaptation algorithm, described in Section 5.3.3 – and compared them to the domain adaptation derived from regularized logistic regression proposed by Chelba and Acero (2006), described in Section 5.3.1.

5.3.1 Logistic Regression for Domain Adaptation Setting with Modified Regularization Term

The method proposed in Chelba and Acero (2006) for maximum entropy models involves modifying the optimization function. First, this method learns a model for the source domain, θ_S, by using the training instances from the source domain, (X_S, Y_S), where $X_S \in \mathbb{R}^{m_S \times n}$ and $Y_S \in \mathcal{Y}^{m_S}$ (note that the subscripts indicate the domain, with S for the source, and T – in the subsequent equations – for the target).

$$\theta_S = \arg\max_{\theta_S} \left[l(\theta_S) - \lambda \|\theta_S\|^2 \right] \tag{5.11}$$

Then, using the source model to constrain the target model, learn a model of the target domain, θ_T, by using the training instances from the target domain, (X_T, Y_T), where $X_T \in \mathbb{R}^{m_T \times n}$ and $Y_T \in \mathcal{Y}^{m_T}$, but with the following modified optimization function:

$$\theta_T = \arg\max_{\theta_T} \left[l(\theta_T) - \lambda \|\theta_T - \theta_S\|^2 \right] \tag{5.12}$$

5.3.2 Supervised Domain Adaptation Derived from Regularized Logistic Regression

The first method proposed uses a convex combination of two logistic regression classifiers – one trained on the source data, and the other trained on the target data, as shown in Figure 5.18. First, a model for the source domain and a model for the target domain are learned, using the training instances from the source domain, (X_S, Y_S) and from the target domain, (X_T, Y_T), respectively:

$$\theta_S = \arg\max_{\theta_S} \left[l(\theta_S) - \lambda \|\theta_S\|^2 \right] \tag{5.13}$$

$$\theta_T = \arg\max_{\theta_T} \left[l(\theta_T) - \lambda \|\theta_T\|^2 \right] \tag{5.14}$$

Then, using these models, the posterior probability for every instance x from the test set of the target domain was approximated as a normalized convex combination of the posterior probabilities for the source and target domains:

$$P(y \mid x; \alpha) \propto (1 - \alpha) \cdot P_S(y \mid x; \theta_S) + \alpha \cdot P_T(y \mid x; \theta_T) \tag{5.15}$$

where $\alpha \in [0, 1]$ is a parameter that shifts the weight from source domain to target domain depending on the distance between these domains, and the amount of target data available.

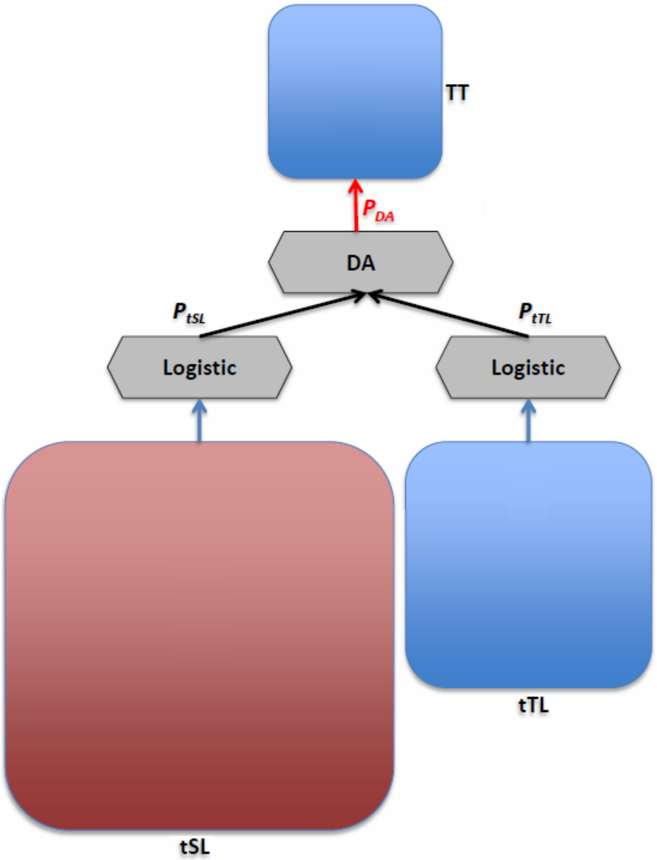


Figure 5.18: Supervised domain adaptation classifiers derived from regularized logistic regression.

5.3.3 Semi-supervised Domain Adaptation Derived from Regularized Logistic Regression

The second method proposed is a modified version of the algorithm proposed in Section 5.2. A couple of changes were made to that algorithm. First, the regularized logistic regression was used as the supervised classifier, instead of the naïve Bayes classifier, and second, three variants of incorporating the target unlabeled data were evaluated, as described below. To make the results comparable to the ones of the algorithm proposed Section 5.2, no additional changes were made (besides the two mentioned above) – i.e., features from the source domain are filtered out, even though the same was not done with the other algorithms, as the latter do not use target unlabeled data. Note that there are two main differences between this classifier and the one proposed in Section 5.3.2: this is an iterative algorithm, using the expectation-maximization approach, and, this algorithm uses the target unlabeled data when learning a model.

In the first step, the domain specific features from the source domain are filtered out, and only the top ranking features are kept, ranked with $f(x_j)$, that have similar mutual information in both the source and target labeled data sets:

$$f(x_j) = \frac{I_{tSL}(x_j; y) \cdot I_{tTL}(x_j; y)}{|I_{tSL}(x_j; y) - I_{tTL}(x_j; y)| + \rho} \tag{5.16}$$

where $I(x_j; y)$ is the mutual information between feature x_j and class y in the training source labeled dataset, tSL, and in the training target labeled data, tTL, and ρ is a parameter used to prevent division by zero. For these experiments ρ is set to $4.9E^{-324}$, for minimal influence on the value of $f(x_j)$.

Once the domain specific features are filtered out from the source domain, the posterior probabilities are estimated from the training source labeled data, p_{tSL}, and from the training target labeled data, p_{tTL}, using the regularized logistic regression classifier. Then, with these probabilities the instances from the training target unlabeled dataset are labeled:

$$P = \alpha \cdot P_{tTL} + (1 - \alpha) \cdot P_{tSL} \tag{5.17}$$

where $\alpha \in (0, 1)$ is a parameter that determines how much weight is assigned to source and target instances. Three variants are used to label the instances from training unlabeled dataset. In one variant, soft labels are assigned to all instances. In another variant, hard labels are assigned to a set number of instances, proportional to the prior, the ones with the most confident predictions. Since for the splice site prediction, the datasets have a ratio of positive to negative of 1:99, at each iteration 100 instances are hard-labeled, 1 positive and 99 negative. In the other variant, at each iteration the most confident instances are labeled with hard labels, and the remaining ones with soft labels.

Then the instances from the training target unlabeled dataset are incorporated, a new classifier is built, the labels for the soft-labeled instances are predicted, and loop until the labels assigned to unlabeled data no longer change:

$$P = \alpha \cdot P_{tTL} + (1 - \alpha) \cdot [\gamma \cdot P_{tTU} + (1 - \gamma) \cdot P_{tSL}] \tag{5.18}$$

where $\gamma = \min(\tau \cdot \beta, 1)$, and τ is the iteration number, and $\beta \in (0, 1)$ is a parameter that splits the weight between the source and target unlabeled instances. Note that when checking for convergence the hard labels assigned to target unlabeled instances are used (i.e., for the purpose of checking for convergence, hard labels are temporarily assigned to all instances from the training target unlabeled dataset, which are used for comparison between iterations). The algorithm is shown in Figure 5.19 and its pseudo-code is listed in Algorithm 4.

5.3.4 Experimental Setup

To find the optimal parameters' values the three folds of 100,000 instances from the target domain were used as validation sets. A grid search was perfomed for λ, using the baseline, supervised logistic classifier, with $\lambda = 10^x, x \in \{-8, -6, \ldots, 4\}$, trained with data from source and target domains. For these datasets the best results were obtained when $\lambda = 1,000$. Therefore, for the proposed algorithm λ_S and λ_T were set to 1,000, and a grid search was performed for δ with values from $\{0.1, 0.2, \ldots, 0.9\}$, whereas for the method proposed by Chelba and Acero (2006) λ_S was set to 1,000 and a grid search was performed for λ_T with $\lambda_T = 10^x, x \in \{-8 - 7, \ldots, 4\}$. For the method in (Chelba and Acero 2006) λ_T was optimized, as λ_T controls the trade-off between source and target parameters, and thus it is similar to the δ parameter for the first proposed method. For the second proposed method a grid search was performed for $\alpha, \beta \in \{0.2, 0.4, 0.6, 0.8\}$ and $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ for generalizable features kept in the source domain.

For the domain adaptation setting the classifiers were trained on source and target data, whereas for the baseline classifiers, the supervised logistic regression, in one setting the

Algorithm 4 Outline of the semi-supervised domain adaptation with logistic regression, with hard and soft labels $\mathrm{SS^{h+s}DA^{loc}_{LR}}$, with hard labels only $\mathrm{SS^{h}DA^{loc}_{LR}}$, and with soft labels only $\mathrm{SS^{s}DA^{loc}_{LR}}$.

```
1: Remove domain specific features from the source dataset, using Equation (5.16).
2: Initialize $TU_s = TU$ and $TU_h = \emptyset$, where $TU_s$ is the set of target unlabeled instances
   with soft labels assigned, $TU_h$ is the set of target unlabeled instances with hard labels
   assigned, and $TU$ is the set of target unlabeled instances passed to the algorithm.
3: Train a classifier using Equation (5.17).
4: Assign labels to the unlabeled instances from the target domain using this classifier.
   The labels assigned are either: soft and hard labels, hard labels only, or soft labels only.
   Any instances assigned hard labels are removed from $TU_s$ and added to $TU_h$.
5: while labels assigned to instances in $TU_s$ change do
6:     M-step: Train a classifier using Equation (5.18)., i.e., also use the instances from
       the target unlabeled dataset that were labeled in steps 3 and 6.
7:     E-step: Same as step 3.
8: end while
9: Use classifier trained using Equation (5.18), on new target instances.
```

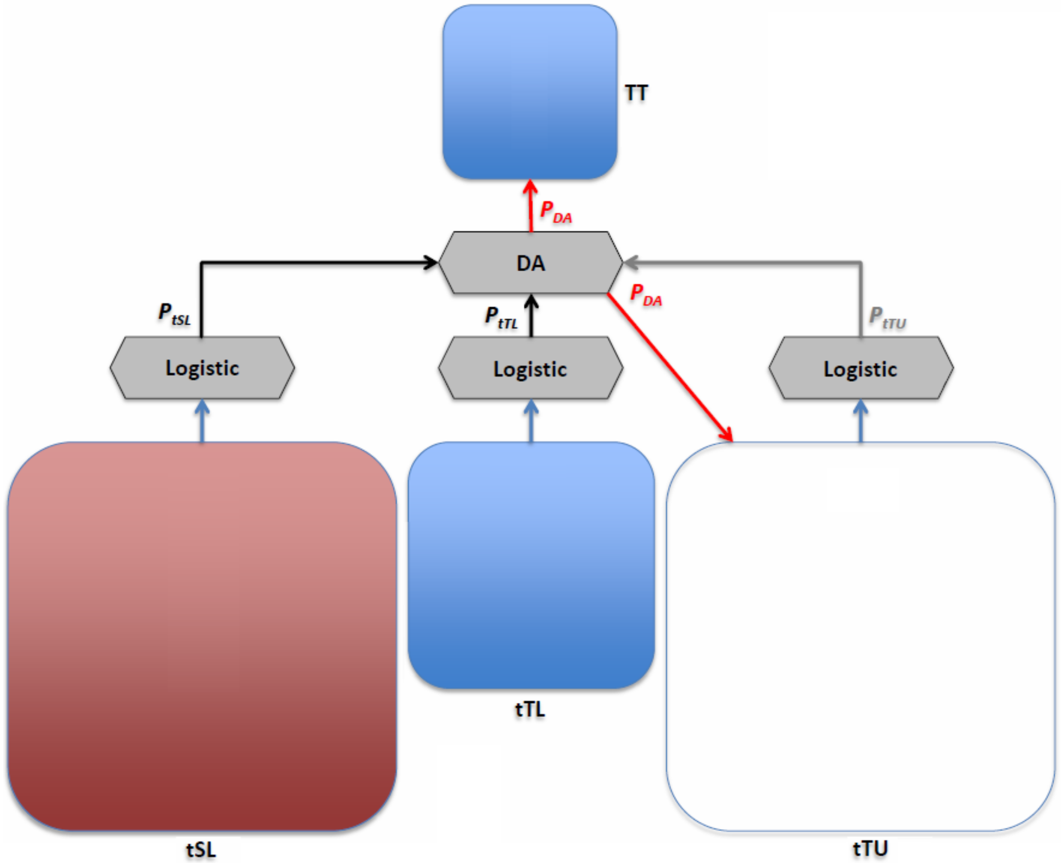


Figure 5.19: Semi-supervised domain adaptation classifiers derived from regularized logistic regression.

classifier was trained on source, and in another setting the classifier was trained on each of the labeled target data set sizes: 2,500, 6,500, 16,000, and 40,000. To evaluate the classifiers they were tested on the test target data from the corresponding fold. It is expected that the results of the baseline, logistic regression classifier trained on each of the target labeled data sets to be the lower bound for the proposed method trained on the source data and that corresponding target labeled data, since it is believed that adding data from a related organism should produce a better classifier.

This experimental setup was used to answer several questions specific to this classifier, in addition to the general questions listed in Section 5.1.2 – namely, how does the performance of the classifier vary with:

Q11 The influence of the following factors on the performance of the classifier:

(a) The weight assigned to the target data.

(b) Variant used to assign labels to instances from the training target unlabeled dataset: using soft labels only, hard labels only, or a combination of both.

Q12 The performance of the domain adaptation classifiers derived from the supervised logistic regression classifier (the method proposed in (Chelba and Acero 2006), and these proposed methods), compared to other domain adaptation classifiers for the task of splice site prediction, i.e., an SVM classifier (Schweikert et al. 2008) and the naïve Bayes classifier proposed in Section 5.2.

Table 5.4: auPRC values for the minority class for four target organisms based on the number of labeled target instances used for training: 2,500, 6,500, 16,000, and 40,000. The LR_{tSL} and LR_{tTL} are the baseline supervised logistic regression classifiers trained on source and target labeled data, respectively. The $\text{S\_DA}^{\text{loc}}_{\text{LR\_reg}}$ is the domain adaptation classifier proposed in (Chelba and Acero 2006). The $\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$ is the best overall semi-supervised domain adaptation proposed in the previous section. $\text{SS}^{\text{s}}\text{DA}^{\text{loc}}_{\text{LR}}$ is the proposed semi-supervised domain adaptation algorithm that assigns only soft labels to instances from target unlabeled dataset, $\text{SS}^{\text{h+s}}\text{DA}^{\text{loc}}_{\text{LR}}$ assigns at each iteration hard labels to the most confident predictions and soft labels to the remaining instances, and $\text{SS}^{\text{h}}\text{DA}^{\text{loc}}_{\text{LR}}$ assigns at each iteration hard labels to the most confident predictions. For comparison with these classifiers the values for the best overall classifier in (Schweikert et al. 2008), $\text{SVM}_{S,T}$, are shown (listed in these subtables as S_DA_{SVM}). The best average value for each target dataset size is shown in bold.

(a) *C.remanei*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	77.63±1.37			
	LR_{tTL}	31.07±8.72	54.20±3.97	65.73±2.76	72.93±1.70
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	77.64±1.39	77.75±1.25	77.88±1.42	78.10±1.15
	$\text{S\_DA}^{\text{loc}}_{\text{LR\_reg}}$	16.30±7.70	40.87±3.26	49.07±0.93	58.37±2.63
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	59.18±1.17	63.10±1.23	63.95±2.08	63.80±1.41
	$\text{SS}^{\text{s}}\text{DA}^{\text{loc}}_{\text{LR}}$	77.63±1.11	77.76±0.98	77.86±1.10	78.02±0.85
	$\text{SS}^{\text{h}}\text{DA}^{\text{loc}}_{\text{LR}}$	77.39±1.03	77.58±0.88	77.80±1.07	77.89±0.81
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc}}_{\text{LR}}$	77.65±1.19	77.74±0.92	77.87±1.16	78.04±0.85
	S_DA_{SVM}	77.06±2.13	77.80±2.89	77.89±0.29	79.02±0.09
1- + 3-mers	LR_{tSL}	81.37±2.27			
	LR_{tTL}	26.93±9.91	55.26±2.21	68.30±1.91	77.33±2.78
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	81.39±2.30	**81.47**±2.19	**81.78**±2.08	**82.61**±2.00
	$\text{S\_DA}^{\text{loc}}_{\text{LR\_reg}}$	2.30±1.05	14.50±4.68	40.10±3.72	63.53±7.10
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	45.29±2.62	72.00±4.16	74.83±4.32	77.07±4.45
	$\text{SS}^{\text{s}}\text{DA}^{\text{loc}}_{\text{LR}}$	81.05±1.82	81.04±1.39	67.95±1.53	76.97±2.26
	$\text{SS}^{\text{h}}\text{DA}^{\text{loc}}_{\text{LR}}$	80.66±1.77	79.84±1.17	67.12±1.49	77.46±2.47
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc}}_{\text{LR}}$	**81.40**±1.89	81.42±1.84	81.75±1.74	82.54±1.64

Table 5.4: (Cont.)

(b) *P.pacificus*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	64.20±1.91			
	LR_{tTL}	29.87±3.58	49.03±4.90	59.93±2.74	69.10±2.25
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR}}$	64.70±1.85	65.31±2.10	66.76±0.89	70.18±2.12
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR\_reg}}$	18.00±3.83	32.73±2.69	40.73±4.30	55.73±1.62
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc:filtered}}_{\mathrm{NB}}$	45.32±2.68	49.82±2.58	52.09±2.04	54.62±1.51
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	**66.11**±1.50	66.36±1.60	67.32±0.72	70.19±1.70
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	63.98±1.66	64.70±1.77	66.31±0.61	69.95±1.72
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	64.82±1.46	65.46±1.92	67.03±0.85	70.20±1.70
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	64.72±3.75	**66.39**±0.66	68.44±0.67	71.00±0.38
1- + 3-mers	LR_{tSL}	62.37±0.84			
	LR_{tTL}	28.40±4.49	49.67±2.83	62.97±3.32	74.60±2.85
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR}}$	64.18±1.10	65.49±1.84	**69.76**±2.08	**75.82**±2.00
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR\_reg}}$	4.37±1.76	14.50±4.86	38.23±6.54	63.70±5.28
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc:filtered}}_{\mathrm{NB}}$	20.21±1.17	53.29±3.08	62.33±3.60	69.88±4.04
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	64.47±1.23	65.40±1.51	62.66±2.57	74.09±2.39
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	61.16±1.33	63.13±1.92	60.66±3.53	74.64±2.60
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	64.55±1.05	65.59±1.68	68.71±1.29	74.81±1.62

Table 5.4: (Cont.)

(c) *D.melanogaster*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	35.87±2.32			
	LR_{tTL}	19.97±3.48	31.80±3.86	42.37±2.15	50.53±1.80
	$\mathrm{S\_DA}_{\mathrm{LR}}^{\mathrm{loc}}$	39.70±2.82	42.19±3.41	49.72±2.01	53.43±0.89
	$\mathrm{S\_DA}_{\mathrm{LR\_reg}}^{\mathrm{loc}}$	11.33±1.36	22.80±2.60	27.30±3.92	42.67±0.76
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	33.31±3.71	36.43±2.18	40.32±2.04	42.37±1.51
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	42.61±1.62	44.44±1.93	49.80±1.59	53.63±0.80
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	**48.02**±1.10	**47.24**±1.27	50.18±1.73	53.76±0.80
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	41.70±2.01	44.15±2.00	49.76±1.61	53.64±0.79
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	40.80±2.18	37.87±3.77	**52.33**±0.91	**58.17**±1.50
1- + 3-mers	LR_{tSL}	32.23±2.76			
	LR_{tTL}	15.07±4.11	28.30±5.45	44.67±3.23	38.43±32.36
	$\mathrm{S\_DA}_{\mathrm{LR}}^{\mathrm{loc}}$	37.24±2.20	40.93±3.79	50.54±3.91	45.89±22.25
	$\mathrm{S\_DA}_{\mathrm{LR\_reg}}^{\mathrm{loc}}$	3.40±1.82	8.37±2.48	21.20±2.85	26.50±22.44
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{NB}}^{\mathrm{loc:filtered}}$	25.83±2.35	32.58±5.83	39.10±1.82	47.49±3.44
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	37.00±2.02	40.51±3.05	48.46±1.35	47.11±13.69
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	33.29±2.48	37.57±3.69	47.26±1.87	41.96±21.19
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}_{\mathrm{LR}}^{\mathrm{loc}}$	37.15±2.03	40.80±3.03	50.82±2.70	48.35±15.26

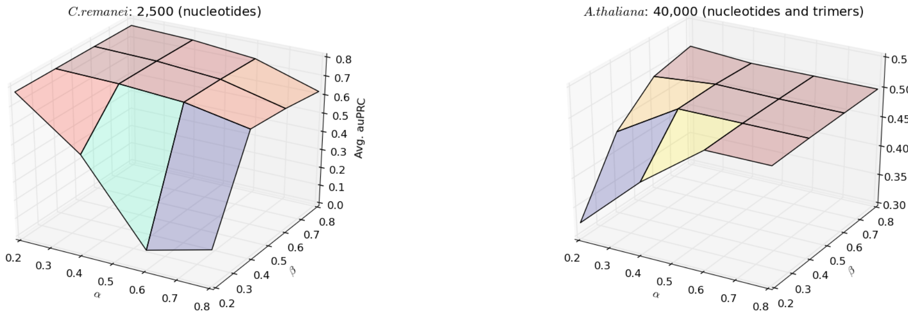
Table 5.4: (Cont.)

(d) *A.thaliana*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	16.93±0.21			
	LR_{tTL}	13.87±2.63	26.03±3.29	38.43±6.18	49.33±4.07
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR}}$	20.67±0.58	27.19±1.30	40.56±3.26	49.75±2.82
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR\_reg}}$	8.50±2.08	17.93±4.72	23.30±2.35	39.10±4.97
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc:filtered}}_{\mathrm{NB}}$	18.46±1.13	25.04±0.72	31.47±3.56	36.95±3.39
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	25.84±0.48	**32.50**±1.17	**43.03**±3.75	50.59±3.50
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	**29.87**±0.73	29.54±1.27	41.30±4.15	50.24±3.68
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	23.43±0.28	32.18±1.28	42.65±3.74	50.61±3.52
	$\mathrm{S\_DA}_{\mathrm{SVM}}$	24.21±3.41	27.30±1.46	38.49±1.59	49.75±1.46
1- + 3-mers	LR_{tSL}	14.07±0.31			
	LR_{tTL}	8.87±1.84	21.10±4.45	38.53±8.08	49.77±2.77
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR}}$	16.42±1.20	26.44±2.49	41.35±6.49	**50.83**±2.28
	$\mathrm{S\_DA}^{\mathrm{loc}}_{\mathrm{LR\_reg}}$	2.50±0.10	8.27±1.60	20.03±3.36	30.27±2.57
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc:filtered}}_{\mathrm{NB}}$	3.99±0.43	13.96±2.42	33.62±6.31	43.20±3.78
	$\mathrm{SS}^{\mathrm{s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	16.50±0.68	27.00±2.30	40.86±4.58	49.67±2.36
	$\mathrm{SS}^{\mathrm{h}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	13.15±0.34	21.63±2.04	39.50±3.87	49.49±2.16
	$\mathrm{SS}^{\mathrm{h+s}}\mathrm{DA}^{\mathrm{loc}}_{\mathrm{LR}}$	16.64±1.11	27.34±2.25	41.76±5.21	50.57±2.04

5.3.5 Results and Discussion

Table 5.4 and Figure 5.21 show the average auPRC values for the minority class. Based on these results, the following observations can be made:



(a) Low α and high β.

(b) High α and high β.

Figure 5.20: The second proposed method produced the most accurate classification with low values for α and high values for β when the target domain was close to the source domain (e.g., *C.remanei*). As the distance between the source and target domains increased (e.g., *A.thaliana*), the classifier performed best with increasing values for α and high values for β.

A1 **Features**: the proposed classifiers performed better with nucleotide and 3-mer features, when the source and target domains are closely related and the classifier has more target labeled data available. However, as the distance between the source and target domains increases, the proposed algorithm performs better with nucleotide features when there is little target labeled data. This conforms with the results in Section 5.2: since 3-mers generate a sparse set of features, they lead to decreased classification accuracy when there are a small number of target training instances.

A2 **Amount of target labeled training data**: the more target training data used by the classifier the better the classifier performs. This makes sense, as more sample data describes more closely the distribution.

A3 **Distance between domains**: as the distance between the source and target domains increases the contribution of the source data decreases. It is interesting to note though that based on these results the splice site prediction problem seems to be more difficult for more complex organisms. For all dataset sizes and all algorithms evaluated there is a common trend of decreasing auPRC values as the complexity of the organisms increases, from *C.remanei*, *P. pacificus*, *D.melanogaster*, to *A.thaliana*. This is a major reason that helps explain the decreased auPRC values for all classifiers, for

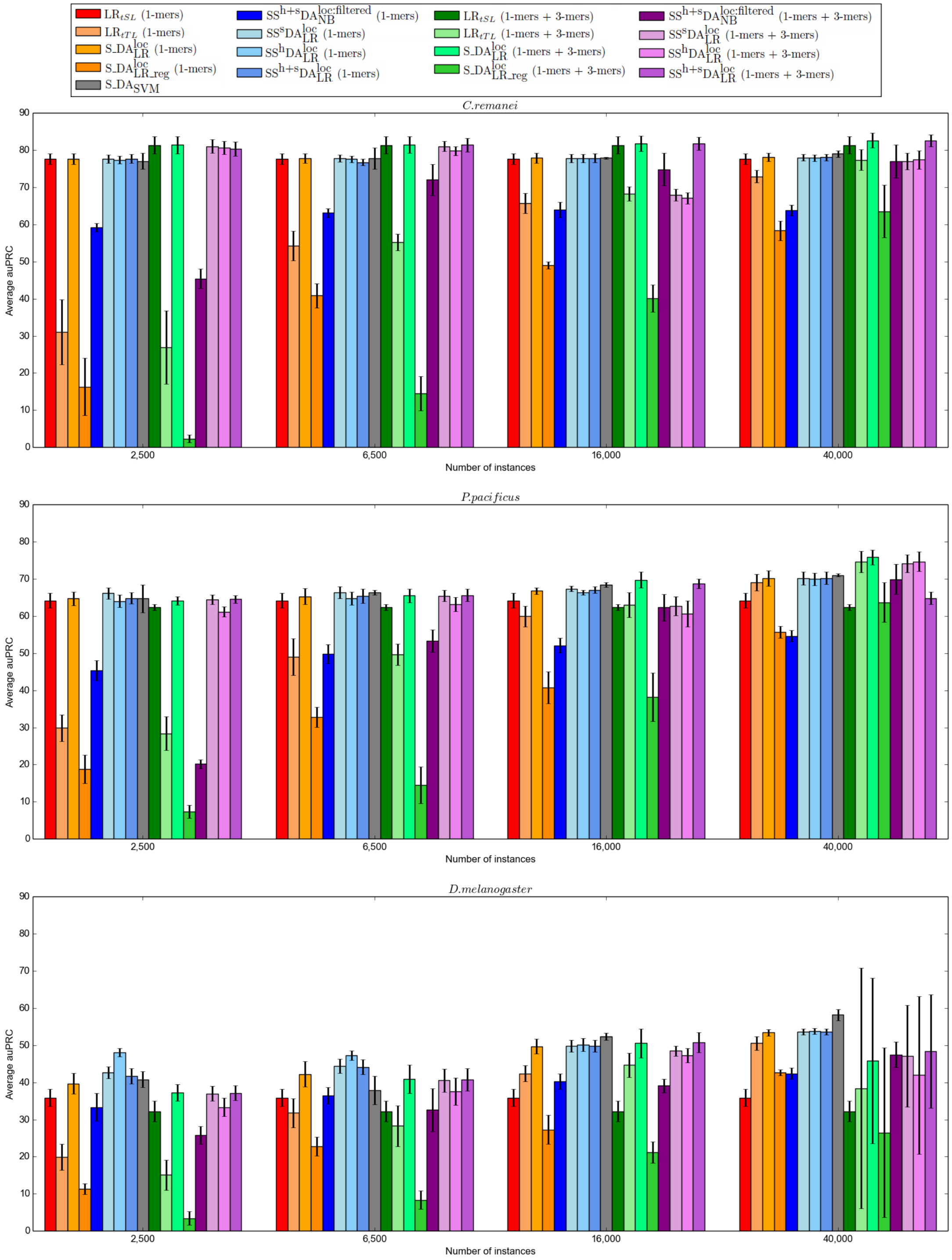


Figure 5.21: Results of the domain adaptation classifiers derived from regularized logistic regression.

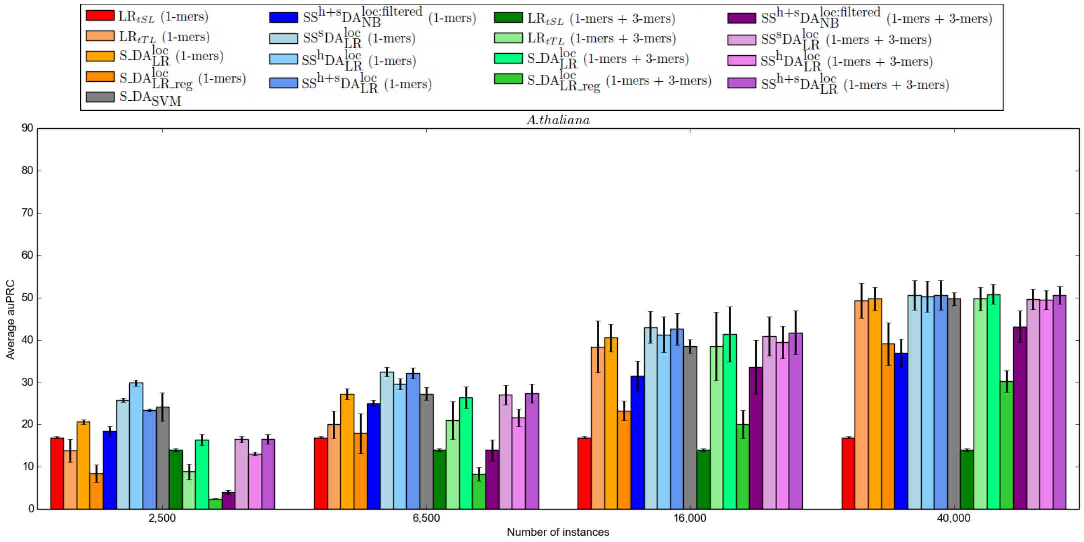


Figure 5.16: (Cont.)

these organisms, respectively, i.e., in general auPRC for *C.remanei* $>$ *P. pacificus* $>$ $\ldots >$ *A.thaliana*.

A11 Factors that influence the performance of the classifier:

(a) **Weight assigned to target data**: intuitively, for the first proposed method in this section, the expectation is for δ to be closer to one when the source and target domain are more distantly related, and closer to zero otherwise. The results conform with this intuition, with δ between 0.1 and 0.6 for *C.remanei*, between 0.7 and 0.8 for *P.pacificus*, between 0.8 and 0.9 for *D.melanogaster*, and 0.9 for *A.thaliana*.

For the second proposed method in this section, the expectation is for α to be small for closely related source and target domains, since there is more data available in the source domain; as the distance between domains increases, the expectation is that the best results are obtained with increasing values for α, which assign more weight to the target labeled data. For β the expectation is that the best results are obtained for high β values, as after a few iterations there should be enough confidently labeled data in the target domain. The results confirm this intuition, as shown in Figure 5.20.

(b) **Type of labels used for instances from target unlabeled dataset**: in most cases using a combination of hard and soft labels produced better results than using soft labels only, which is in turn better than using hard labels only. It is interesting to note that when using nucleotides and 3-mers as features, the combination of hard and soft labels produced best results. On the other hand, when using nucleotides as features (i) and there is enough target labeled data (40,000 instances), the best results are obtained when using the combination of labels, or (ii) when there is less target labeled data the results are best when using soft labels only, except for the only three cases for which using hard labels only generated best results (when using 2,500 or 6,500 instances with *D.melanogaster*, and 2,500 instances with *A.thaliana*). These results conform with the intuition that hard labels should only be assigned to the most confident instances and the remaining instances should not be discarded but instead should be used with soft labels. For the three cases of best results when using hard labels only, one hypothesis is that the most confident predictions had probabilities close to $y = (1, 0)$ for positive and $y = (0, 1)$ for negative instances and therefore assigning the nearest hard label did not skew the classifier by much. Similarly, when the features are nucleotides and there is not enough target labeled data, some of the most confident predictions had probabilities that were not close to (1,0), or (0,1), respectively, and assigning hard labels to these instances skewed the classifier leading to worse accuracy as opposed to assigning soft labels only.

A12 In terms of performance, the method proposed by Chelba and Acero (2006) produced worse results than the supervised logistic regression classifier trained on the target data. These poor results are due to modified optimization function of this method, which constrains the values of the parameters for the target domain, θ_T, to be close to the values of the parameters for the source domain, θ_S. In addition, this method performed worse than the domain adaptation naïve Bayes classifier proposed in Section 5.2, except for two cases (when using nucleotides as features, the target domains are *D.melanogaster*, and *A.thaliana*, and the algorithms are trained on 40,000 target instances).

The first method proposed in this section produced better average results than the supervised logistic classifier trained on either the source or the target domain in *every* case of the 16 evaluated. This confirms the hypothesis that augmenting a small labeled dataset from the target domain with a large labeled dataset from a closely related source domain improves the accuracy of the classifier. In addition, this method outperformed the domain adaptation naïve Bayes classifier proposed in Section 5.2, as well as the method proposed in (Chelba and Acero 2006) in every case, and outperformed the best overall domain adaptation SVM classifier proposed in (Schweikert et al. 2008) in 9 out of the 16 cases.

The second method proposed in this section produced better average results with at least one of its three labeling variants than the supervised logistic classifier trained on either the source or the target domain in all cases evaluated. It also outperformed the domain adaptation naïve Bayes classifier proposed in Section 5.2, as well as the method proposed in (Chelba and Acero 2006) in every case, and outperformed the best overall domain adaptation SVM classifier proposed in (Schweikert et al. 2008) in 7 out of the 16 cases.

Based on these results the first method proposed in this section should be used instead of the domain adaptation S_DA$_{\text{SVM}}$ classifier when the source and target domains are closely related, or when there is quite a bit of labeled data for the target domain, otherwise, the second method proposed in this section should be used when there is little labeled data, and the S_DA$_{\text{SVM}}$ algorithm proposed in (Schweikert et al. 2008) in the remaining three cases, namely, for *D.melanogaster* when there's plenty of labeled data (16,000 or 40,000 instances), and for *P.pacificus* when there's some labeled data (6,500 instances).

5.4 Domain Adaptation with Supervised Classifiers

Let the set of independently generated training instances be represented by $X \in \mathbb{R}^{m \times n}$ and their corresponding labels by $Y \in \mathcal{Y}^m$, $\mathcal{Y} = \{0, 1\}$, where m is the number of training instances and n is the number of features.

Given a set of training instances from the source domain, $\mathcal{D}_S = (X_S, Y_S)$, where $X_S \in \mathbb{R}^{m_S \times n}$ and $Y_S \in \mathcal{Y}^{m_S}$, and a set of training instances from the target domain, $\mathcal{D}_T = (X_T, Y_T)$, where $X_T \in \mathbb{R}^{m_T \times n}$ and $Y_T \in \mathcal{Y}^{m_T}$, create an empty dataset $\mathcal{D} = (X, Y)$, where $X \in \mathbb{R}^{(m_S + m_T) \times n}$ and $Y \in \mathcal{Y}^{m_S + m_T}$. For each instance $(x^i, y^i) \in \mathcal{D}_S$ multiply its weight by w_S, then add this instance to the new dataset, $\mathcal{D}$. Similarly, for each instance $(x^i, y^i) \in \mathcal{D}_T$ multiply its weight by w_T, then add it to the new dataset, $\mathcal{D}$. Then, train a supervised classifier on this combined dataset, $\mathcal{D}$, as shown in Figure 5.17. The following two supervised classifiers were used: the WEKA implementations of the regularized logistic regression (Le Cessie and Van Houwelingen 1992), and naïve Bayes (John and Langley 1995).

5.4.1 Experimental Setup

To find the optimal parameters' values a grid search was performed for $w_S, w_T \in \{0.1, 0.2, \ldots, 1\}$.

As baselines, the naïve Bayes and the logistic regression with regularized parameters classifiers were used, trained on either 100,000 from *C.elegans*, or one of the three folds of 2,500, 6,500, 16,000, or 40,000 from the target organisms, and tested them on the corresponding fold for that organism. It is expected that the results of the baseline classifiers will be the lower bound for this proposed method, as adding data from a related organism should improve the accuracy of the classifier. Note, that whenever the logistic regression classifier was used, for baselines or for this proposed method, the ridge parameter was set

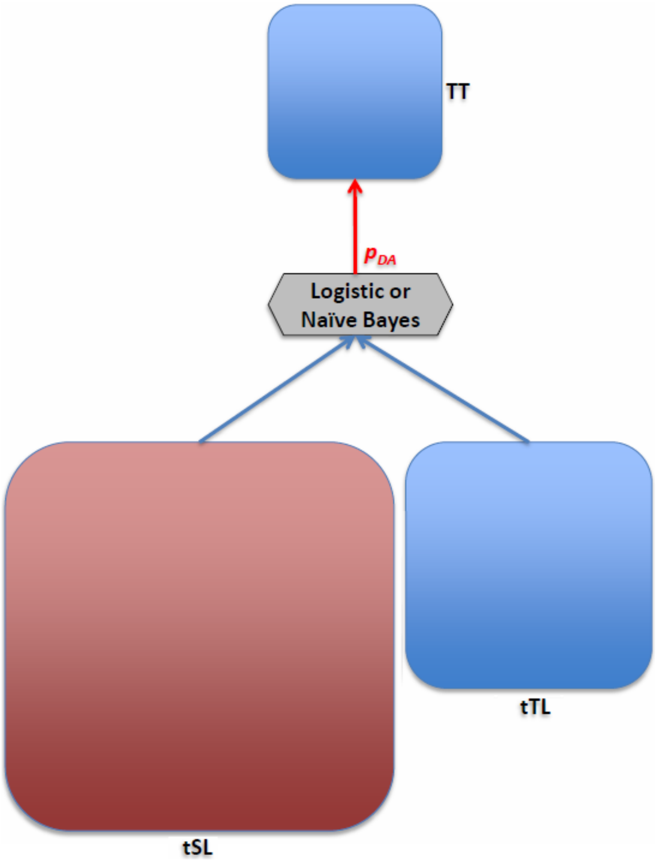


Figure 5.17: Domain adaptation with supervised classifiers.

to 1,000, as this value led to the best results in Section 5.3.

This experimental setup was used to answer several questions specific to this classifier, in addition to the general questions listed in Section 5.1.2 – namely, how does the performance of the classifier vary with:

Q13 How this proposed method (when using naïve Bayes or regularized logistic regression) compares to other domain adaptation classifiers for the task of splice site prediction, namely, the SVM classifier proposed by Schweikert et al. (2008), the naïve Bayes classifier proposed in Section 5.2, and the regularized logistic regression proposed in Section 5.3.

Table 5.5: auPRC values for the minority (i.e., positive) class for four target organisms based on the number of labeled target instances used for training: 2,500, 6,500, 16,000, and 40,000. The LR$_{tSL}$ and LR$_{tTL}$ are the baseline supervised logistic regression classifiers trained on source and target labeled data, respectively. LR$_{tSL+tTL}$ is the proposed supervised domain adaptation classifier using the supervised logistic regression classifier. The S_DA$^{\text{loc}}_{\text{LR}}$ is the supervised domain adaptation classifier proposed in the previous section. S_DA$_{\text{SVM}}$ is the best overall classifier in (Schweikert et al. 2008), SVM$_{S,T}$. The NB$_{tSL}$ and NB$_{tTL}$ are the supervised naïve Bayes classifiers trained on source and target labeled data, respectively. NB$_{tSL+tTL}$ is the proposed supervised domain adaptation classifier using the supervised naïve Bayes classifier. SS$^{\text{h+s}}$DA$^{\text{loc:filtered}}_{\text{NB}}$ is the best overall semi-supervised domain adaptation classifier in Section 5.2. The best average values for each type of features used is shown in bold font.

(a) *C.remanei*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR$_{tSL}$	77.63±1.37			
	LR$_{tTL}$	31.07±8.72	54.20±3.97	65.73±2.76	72.93±1.70
	LR$_{tSL+tTL}$	77.65±1.34	77.88±1.16	78.32±1.29	79.00±0.97
	S_DA$^{\text{loc}}_{\text{LR}}$	77.64±1.39	77.75±1.25	77.88±1.42	78.10±1.15
	NB$_{tSL}$	63.77±1.30			
	NB$_{tTL}$	23.42±7.39	45.44±4.01	54.57±2.63	59.68±1.62
	NB$_{tSL+tTL}$	75.49±1.39	75.56±1.46	75.63±1.45	75.82±1.32
	SS$^{\text{h+s}}$DA$^{\text{loc:filtered}}_{\text{NB}}$	59.18±1.17	63.10±1.23	63.95±2.08	63.80±1.41
	S_DA$_{\text{SVM}}$	77.06±2.13	77.80±2.89	77.89±0.29	79.02±0.09
1- + 3-mers	LR$_{tSL}$	81.37±2.27			
	LR$_{tTL}$	26.93±9.91	55.26±2.21	68.30±1.91	77.33±2.78
	LR$_{tSL+tTL}$	**81.40**±2.25	**81.73**±1.90	**82.62**±2.28	**83.57**±1.76
	S_DA$^{\text{loc}}_{\text{LR}}$	81.39±2.30	81.47±2.19	81.78±2.08	82.61±2.00
	NB$_{tSL}$	77.67±2.24			
	NB$_{tTL}$	22.94±4.37	58.39±3.94	68.40±3.37	75.75±1.32
	NB$_{tSL+tTL}$	81.11±0.73	81.38±0.34	81.51±0.87	82.73±0.52
	SS$^{\text{h+s}}$DA$^{\text{loc:filtered}}_{\text{NB}}$	45.29±2.62	72.00±4.16	74.83±4.32	77.07±4.45

Table 5.5: (Cont.)

(b) *P.pacificus*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	64.20±1.91			
	LR_{tTL}	29.87±3.58	49.03±4.90	59.93±2.74	69.10±2.25
	$\text{LR}_{tSL+tTL}$	64.72±1.85	65.63±1.82	67.09±1.29	70.76±2.08
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	64.70±1.85	65.31±2.10	66.76±0.89	70.18±2.12
	NB_{tSL}	49.12±1.58			
	NB_{tTL}	19.22±3.39	37.33±2.65	45.33±2.28	52.84±2.06
	$\text{NB}_{tSL+tTL}$	60.67±1.97	61.96±2.04	63.04±0.33	65.17±2.09
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	45.32±2.68	49.82±2.58	52.09±2.04	54.62±1.51
	S_DA_{SVM}	64.72±3.75	66.39±0.66	68.44±0.67	71.00±0.38
1- + 3-mers	LR_{tSL}	62.37±0.84			
	LR_{tTL}	28.40±4.49	49.67±2.83	62.97±3.32	74.60±2.85
	$\text{LR}_{tSL+tTL}$	64.14±0.83	66.14±0.55	**70.97**±2.03	**76.89**±1.75
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	64.18±1.10	65.49±1.84	69.76±2.08	75.82±2.00
	NB_{tSL}	**67.10**±1.94			
	NB_{tTL}	26.39±3.97	48.54±3.42	59.29±2.80	68.78±1.52
	$\text{NB}_{tSL+tTL}$	64.51±0.70	66.32±0.71	69.29±2.00	72.54±0.42
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	20.21±1.17	53.29±3.08	62.33±3.60	69.88±4.04

5.4.2 Results and Discussion

In Table 5.5 and Figure 5.18 the auPRC averages over three folds and their standard deviations are shown for the four target organisms for:

- This proposed method: $\text{LR}_{tSL+tTL}$ when using the regularized logistic regression and $\text{NB}_{tSL+tTL}$ when using the naïve Bayes classifier.

- Supervised classifiers used as baselines: LR_{tSL} and LR_{tTL} when using the regularized logistic regression classifier trained on source and target data, respectively, and NB_{tSL} and NB_{tTL} when using the naïve Bayes classifier trained on source and target data, respectively.

- The domain adaptation with naïve Bayes classifier proposed in Section 5.2, $\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$. Note that this is the only classifier, from the ones compared, that used the target unlabeled data in addition to the source and target labeled data.

Table 5.5: (Cont.)

(c) *D.melanogaster*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	35.87±2.32			
	LR_{tTL}	19.97±3.48	31.80±3.86	42.37±2.15	50.53±1.80
	$\text{LR}_{tSL+tTL}$	41.35±1.40	43.66±3.20	49.96±2.09	54.02±0.95
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	39.70±2.82	42.19±3.41	49.72±2.01	53.43±0.89
	NB_{tSL}	31.23±1.03			
	NB_{tTL}	14.90±2.80	26.05±4.79	35.21±2.43	39.42±2.90
	$\text{NB}_{tSL+tTL}$	45.43±0.87	47.12±3.86	51.73±1.24	52.74±2.43
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	33.31±3.71	36.43±2.18	40.32±2.04	42.37±1.51
	S_DA_{SVM}	40.80±2.18	37.87±3.77	52.33±0.91	**58.17**±1.50
1- + 3-mers	LR_{tSL}	32.23±2.76			
	LR_{tTL}	15.07±4.11	28.30±5.45	44.67±3.23	38.43±32.36
	$\text{LR}_{tSL+tTL}$	34.97±2.59	37.22±4.30	49.16±5.11	43.03±22.03
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	37.24±2.20	40.93±3.79	50.54±3.91	45.89±22.25
	NB_{tSL}	34.09±2.44			
	NB_{tTL}	13.87±2.97	25.00±5.59	35.28±2.14	45.85±3.32
	$\text{NB}_{tSL+tTL}$	**46.85**±1.41	**50.84**±4.39	**56.57**±2.37	50.15±14.84
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	25.83±2.35	32.58±5.83	39.10±1.82	47.49±3.44

- The domain adaptation with regularized logistic regression proposed in Section 5.3, $\text{S\_DA}^{\text{loc}}_{\text{LR}}$.
- The supervised domain adaptation with SVM classifier proposed by Schweikert et al. (2008), S_DA_{SVM}. Note that this classifier used other features to represent the DNA sequences (i.e., it did not represent them with nucleotides and 3-mers along with their positions).

Based on these results the following observations were made:

A1 **Features**: there is a similar trend for this proposed method as with previous classifiers proposed in Sections 5.2 and 5.3, namely, using simple features (the nucleotides) leads to more accurate classifiers when the source and target domains are distant and there is scarce labeled data in the target domain. Using a combination of simple and complex features (nucleotides and 3-mers) leads to more accurate classifiers when the source

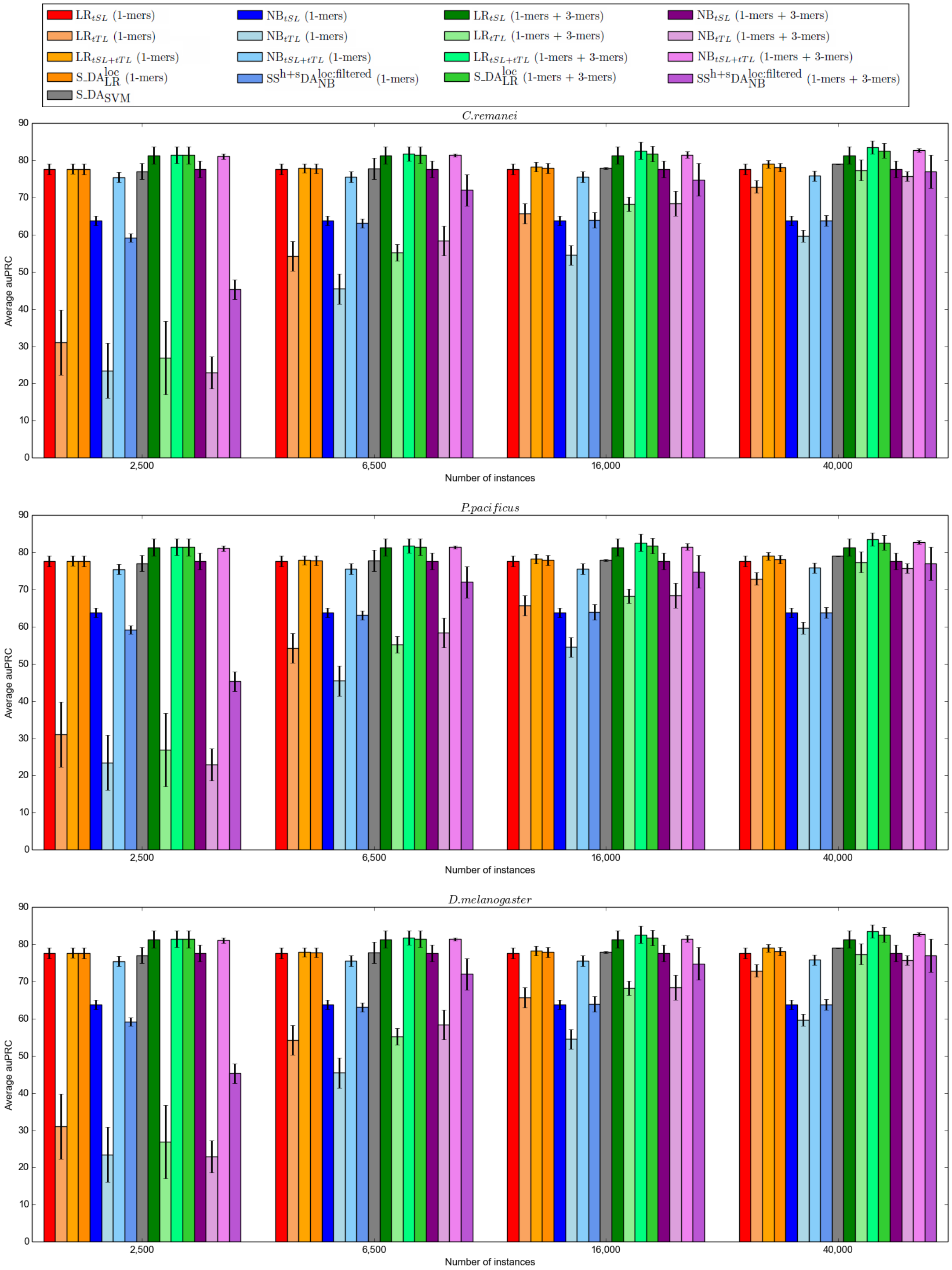


Figure 5.18: Results of the domain adaptation method using supervised classifiers.

Table 5.5: (Cont.)

(d) *A.thaliana*

Features	Classifier	2,500	6,500	16,000	40,000
1-mers	LR_{tSL}	16.93±0.21			
	LR_{tTL}	13.87±2.63	26.03±3.29	38.43±6.18	49.33±4.07
	$\text{LR}_{tSL+tTL}$	22.79±0.92	**31.70**±2.70	41.28±2.64	49.91±2.38
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	20.67±0.58	27.19±1.30	40.56±3.26	49.75±2.82
	NB_{tSL}	11.97±0.23			
	NB_{tTL}	7.21±0.90	17.90±1.93	28.10±4.68	34.82±4.77
	$\text{NB}_{tSL+tTL}$	23.30±1.18	30.97±2.31	39.18±2.79	44.88±3.13
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	18.46±1.13	25.04±0.72	31.47±3.56	36.95±3.39
	S_DA_{SVM}	**24.21**±3.41	27.30±1.46	38.49±1.59	49.75±1.46
1- + 3-mers	LR_{tSL}	14.07±0.31			
	LR_{tTL}	8.87±1.84	21.10±4.45	38.53±8.08	49.77±2.77
	$\text{LR}_{tSL+tTL}$	15.87±0.36	23.65±1.49	39.97±4.39	50.60±2.11
	$\text{S\_DA}^{\text{loc}}_{\text{LR}}$	16.42±1.20	26.44±2.49	41.35±6.49	50.83±2.28
	NB_{tSL}	13.98±0.71			
	NB_{tTL}	3.10±0.35	8.76±1.65	28.21±7.58	40.92±3.78
	$\text{NB}_{tSL+tTL}$	21.62±1.02	27.89±2.19	**43.52**±6.16	**53.33**±3.77
	$\text{SS}^{\text{h+s}}\text{DA}^{\text{loc:filtered}}_{\text{NB}}$	3.99±0.43	13.96±2.42	33.62±6.31	43.20±3.78

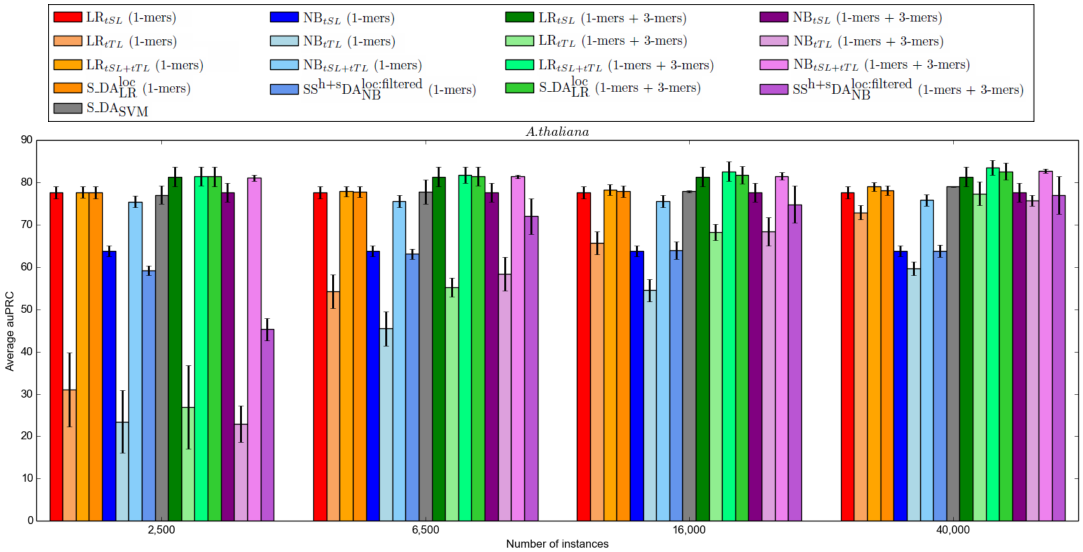


Figure 5.18: (Cont.)

and target domains are closed and there is enough target labeled data. This is expected as 3-mer features are sparser than nucleotide features, and with less labeled data the classifier performs worse with 3-mer features as it does not have enough data to learn an accurate classifier.

A2 **Amount of target labeled data**: as the amount of target labeled data increases the accuracy of this proposed method increases as well, with one exception, though. For *D.melanogaster*, when using nucleotide and 3-mer features, the auPRC decreases as the amount of target labeled data increases from 16,000 to 40,000, regardless of the type of supervised classifier used, naïve Bayes, or regularized logistic regression. It is interesting to note that for this combination of features used and target domain, the auPRC for the regularized logistic regression classifier also decreases when the amount of target labeled data increases from 16,000 to 40,000. This partially explains this exception for this proposed method when using the logistic regression classifier. Another factor, suggested by the large standard deviation, is that the frequency of features is very different between training and test datasets, especially for 3-mers, since using only nucleotide features does not exhibit this behavior.

A3 **Distance between domains**: as the distance between the source and target domains increases, the contribution from the source data decreases, and the accuracy of this method decreases, which is expected.

A11 **Weight assigned to source and target data**: in regards to the weight assigned to the target labeled data, the best results are obtained when w_T is set to one, or close to one. For the weight assigned to the source labeled data, when the domains are closely related the best results are for high values of w_S, but as the distance between domains increases the value for w_S decreases. This is intuitive as it only makes sense to decrease the weight assigned to source data when the distance between domains increases.

A13 In terms of performance, this proposed method produced the best results out of all domain adaptation classifiers compared, when the source and target domains are closely related (for *C.remanei* and *P.pacificus*)), using logistic regression with nucleotide and 3-mer features. It also produced the best results when the domains are distant (for *D.melanogaster* and *A.thaliana*), using naïve Bayes with nucleotide and 3-mer features, in five out of eight cases. This is a similar behavior to the one observed in (Ng and Jordan 2001), namely that a generative classifier performs better than a discriminative one when there is a small amount of training labeled data. For domain adaptation, when the domains are close the source labeled data contributes a lot to the classifier so a discriminative classifier performs better than a generative one. When the domains are distant, the source labeled data contributes less and a generative classifier performs better than a discriminative one. Another case for which this method produced the best results is for very distant domains (*A.thaliana*), using logistic regression with nucleotide features, when there is somewhat scarce target labeled data (6,500 instances). There are only two cases in which another domain adaptation classifier, the SVM proposed by Schweikert et al. (2008), outperformed this proposed method.

Chapter 6

Conclusions and Future Work

This work summarizes the results of several domain adaptation methods proposed, derived from naïve Bayes and logistic regression. These classifiers try to address the lack of or limited amount of labeled data for a target domain, by leveraging the large amount of labeled data from a related domain. These methods are evaluated on two biological problems, protein localization and splice site prediction, and compared them with the domain adaptation classifier derived from the supervised logistic regression classifier, proposed in (Chelba and Acero 2006), the supervised logistic regression and naïve Bayes (as baselines), the and the SVM classifier proposed in (Schweikert et al. 2008).

Several observations were made from the experimental results, such as, in some cases simple features are preferred over complex ones when the latter can lead to sparse representations and decreased accuracy, and vice versa; using more labeled data increases the accuracy of the classifier; and that as the distance between the domains increases the contribution of the source data decreases. In addition, some of the proposed methods performed better than previously proposed methods, recommending them for *ab initio* splice site prediction.

For future work several ways to further increase their accuracy will be explored. For example, balanced subsamples can be created through undersampling, and then training an ensemble of classifiers on these subsamples. In addition, ensembles of classifiers produced by the different methods proposed, can be trained on balanced datasets. Another direction for future work is to combine data from multiple organisms and train a classifier for a target organism, i.e., use multiple source domains. Furthermore, the effectiveness of these proposed methods will be evaluated on other problems that can be addressed in a domain adaptation framework, e.g. text classification problems, sentiment analysis.

Published Contributions not Included in this Work

In addition to the publications included in this work, I also contributed to the following peer-reviewed publications:

1. Inferential considerations for low-count RNA-seq transcripts: a case study on the dominant prairie grass *Andropogon gerardii*, (Raithel et al. 2016).

   Background: Differential expression (DE) analysis of RNA-seq data still poses inferential challenges, such as handling of transcripts characterized by low expression levels. In this study, we use a plasmode-based approach to assess the relative performance of alternative inferential strategies on RNA-seq transcripts, with special emphasis on transcripts characterized by a small number of read counts, so-called low-count transcripts, as motivated by an ecological application in prairie grasses. Big bluestem (*Andropogon gerardii*) is a wide-ranging dominant prairie grass of ecological and agricultural importance to the US Midwest while edaphic subspecies sand bluestem (*A. gerardii* ssp. *Hallii*) grows exclusively on sand dunes. Relative to big bluestem, sand bluestem exhibits qualitative phenotypic divergence consistent with enhanced drought tolerance, plausibly associated with transcripts of low expression levels. Our dataset consists of RNA-seq read counts for 25,582 transcripts (60% of which are classified as low-count) collected from leaf tissue of individual plants of big bluestem (n = 4) and sand bluestem (n = 4). Focused on low-count transcripts, we compare alternative ad-hoc data filtering techniques commonly used in RNA-seq pipelines and assess the inferential performance of recently developed statistical methods for DE analysis, namely DESeq2 and edgeR robust. These methods attempt to overcome the inherently noisy behavior of low-count transcripts by either shrinkage or differential weighting of observations, respectively.

Results: Both DE methods seemed to properly control family-wise type 1 error on low-count transcripts, whereas edgeR robust showed greater power and DESeq2 showed greater precision and accuracy. However, specification of the degree of freedom parameter under edgeR robust had a non-trivial impact on inference and should be handled carefully. When properly specified, both DE methods showed overall promising inferential performance on low-count transcripts, suggesting that ad-hoc data filtering steps at arbitrary expression thresholds may be unnecessary. A note of caution is in order regarding the approximate nature of DE tests under both methods.

Conclusions: Practical recommendations for DE inference are provided when low-count RNA-seq transcripts are of interest, as is the case in the comparison of subspecies of bluestem grasses. Insights from this study may also be relevant to other applications focused on transcripts of low expression levels.

2. A Comparative Analysis between k-mers and Community Detection-based Features for the Task of Protein Classification, (Tangirala et al. 2016).

   Machine learning algorithms are widely used to annotate biological sequences. Low-dimensional informative feature vectors can be crucial for the performance of the algorithms. In prior work, we have proposed the use of a community detection approach to construct low dimensional feature sets for nucleotide sequence classification. Our approach used the Hamming distance between short nucleotide subsequences, called k-mers, to construct a network, and subsequently used community detection to identify groups of k-mers that appear frequently in a set of sequences. Whereas this approach worked well for nucleotide sequence classification, it could not be directly used for protein sequences, as the Hamming distance is not a good measure for comparing short protein *k*-mers. To address this limitation, we extended our prior approach by replacing the Hamming distance with substitution scores. Experimental results in different learning scenarios show that the features generated with the new approach are more informative than *k*-mers.

3. Tools and pipelines for BioNano data: molecule assembly pipeline and FASTA super scaffolding tool, (Shelton et al. 2015).

   Background: Genome assembly remains an unsolved problem. Assembly projects face a range of hurdles that confound assembly. Thus a variety of tools and approaches are needed to improve draft genomes.

   Results: We used a custom assembly workflow to optimize consensus genome map assembly, resulting in an assembly equal to the estimated length of the *Tribolium castaneum* genome and with an N50 of more than 1 Mb. We used this map for super scaffolding the *T. castaneum* sequence assembly, more than tripling its N50 with the program Stitch.

   Conclusions: In this article we present software that leverages consensus genome maps assembled from extremely long single molecule maps to increase the contiguity of sequence assemblies. We report the results of applying these tools to validate and improve a 7x Sanger draft of the *T. castaneum* genome.

4. A Massive Expansion of Effector Genes Underlies Gall-Formation in the Wheat Pest *Mayetiola destructor*, (Zhao et al. 2015).

   Gall-forming arthropods are highly specialized herbivores that, in combination with their hosts, produce extended phenotypes with unique morphologies. Many are economically important, and others have improved our understanding of ecology and adaptive radiation. However, the mechanisms that these arthropods use to induce plant galls are poorly understood. We sequenced the genome of the Hessian fly (*Mayetiola destructor*; Diptera: Cecidomyiidae), a plant parasitic gall midge and a pest of wheat (*Triticum* spp.), with the aim of identifying genic modifications that contribute to its plant-parasitic lifestyle. Among several adaptive modifications, we discovered an expansive reservoir of potential effector proteins. Nearly 5% of the 20,163 predicted gene models matched putative effector gene transcripts present in the *M. destructor* larval salivary gland. Another 466 putative effectors were discovered among the genes that have no sequence similarities in other organisms. The largest known arthropod gene family (family SSGP-71) was also discovered within the effector reservoir. SSGP-71 proteins lack sequence homologies to other proteins, but their structures resemble both ubiquitin E3 ligases in plants and E3-ligase-mimicking effectors in plant pathogenic bacteria. SSGP-71 proteins and wheat Skp proteins interact in vivo. Mutations in different SSGP-71 genes avoid the effector-triggered immunity that is directed by the wheat resistance genes *H6* and *H9*. Results point to effectors as the agents responsible for arthropod-induced plant gall formation.

5. Experimental Study with Real-world Data for Android App Security Analysis using Machine Learning, (Roy et al. 2015).
Although Machine Learning (ML) based approaches have shown promise for Android malware detection, a set of critical challenges remain unaddressed. Some of those challenges arise in relation to proper evaluation of the detection approach while others are related to the design decisions of the same. In this paper, we systematically study the impact of these challenges as a set of research questions (i.e., hypotheses). We design an experimentation framework where we can reliably vary several parameters while evaluating ML-based Android malware detection approaches. The results from the experiments are then used to answer the research questions. Meanwhile, we also demonstrate the impact of some challenges on some existing ML-based approaches. The large (market-scale) dataset (benign and malicious apps) we use in the above experiments represents the real-world Android app security analysis scale. We envision this study to encourage the practice of employing a better evaluation strategy and better designs of future ML-based approaches for Android malware detection.

6. An Evaluation of Self-training Styles for Domain Adaptation on the Task of Splice Site Prediction, (Herndon and Caragea 2015c).

   We consider the problem of adding a large unlabeled sample from the target domain to boost the performance of a domain adaptation algorithm when only a small set of labeled examples are available from the target domain. In particular, we consider the problem setting motivated by the task of splice site prediction. For this task, annotating a genome using machine learning requires a lot of labeled data, whereas for non-model organisms, there is only some labeled data and lots of unlabeled data. With domain adaptation one can leverage the large amount of data from a related model organism, along with the labeled and unlabeled data from the organism of interest to train a classifier for the latter. Our goal is to analyze the three ways of incorporating the unlabeled data – with soft labels only (i.e., Expectation-Maximization), with hard labels only (i.e., self-training), or with both soft and hard labels – for the splice site prediction in particular, and more broadly for a general iterative domain adaptation setting. We provide empirical results on splice site prediction indicating that using soft labels only can lead to better classifier compared to the other two ways.

7. Community Detection-Based Feature Construction for Protein Sequence Classification, (Tangirala et al. 2015).

   Machine learning algorithms are widely used to annotate biological sequences. Low-dimensional informative feature vectors can be crucial for the performance of the algorithms. In prior work, we have proposed the use of a community detection approach to construct low dimensional feature sets for nucleotide sequence classification. Our approach uses the Hamming distance between short nucleotide subsequences, called k-mers, to construct a network, and subsequently uses community detection to identify groups of k-mers that appear frequently in a set of sequences. While this approach worked well for nucleotide sequence classification, it could not be directly used for protein sequences, as the Hamming distance is not a good measure for comparing short protein k-mers. To address this limitation, we extend our prior approach by replacing the Hamming distance with substitution scores. Experimental results in different learning scenarios show that the features generated with the new approach are more informative than k-mers.

8. Twitter Mining for Disaster Response: A Domain Adaptation Approach, (Li et al. 2015).

   Microblogging data such as Twitter data contains valuable information that has the potential to help improve the speed, quality, and efficiency of disaster response. Machine learning can help with this by prioritizing the tweets with respect to various classification criteria. However, supervised learning algorithms require labeled data to learn accurate classifiers. Unfortunately, for a new disaster, labeled tweets are not easily available, while they are usually available for previous disasters. Furthermore, unlabeled tweets from the current disaster are accumulating fast. We study the usefulness of labeled data from a prior *source* disaster, together with unlabeled data from the current *target* disaster to learn *domain adaptation* classifiers for the target. Experimental results suggest that, for some tasks, source data itself can be useful for classifying target data. However, for tasks specific to a particular disaster, domain adaptation approaches that use target unlabeled data in addition to source labeled data are superior.

9. Predicting Protein Localization Using a Domain Adaptation Naïve Bayes Classifier with Burrows Wheeler Transform Features, (Herndon et al. 2014).

   The reduced cost of the next generation sequencing technologies provides opportunities to study non-model organisms. However, one challenge is the large volume of data generated and, thus, the need to use automated approaches to annotate these data. Machine learning algorithms could provide a cost-effective solution but they need lots of labeled data and informative features to represent these data. Our proposed approach addresses both these problems by using a domain adaptation classifier in conjunction with features generated with unsupervised techniques to annotate biological sequence data.

Bibliography

Al-Turaiki, I. M., Mathkour, H., Touir, A., and Hammami, S. (2011). Computational approaches for gene prediction: A comparative survey. In *Informatics Engineering and Information Science*, pages 14–25. Springer.

Arita, M., Tsuda, K., and Asai, K. (2002). Modeling splicing sites with pairwise correlations. *Bioinformatics*, 18(suppl 2):S27–S34.

Baten, A. K., Chang, B. C., Halgamuge, S. K., and Li, J. (2006). Splice site identification using probabilistic parameters and SVM classification. *BMC bioinformatics*, 7(Suppl 5):S15.

Baten, A. K., Halgamuge, S. K., Chang, B., and Wickramarachchi, N. (2007). Biological Sequence Data Preprocessing for Classification: A Case Study in Splice Site Identification. In *Proceedings of the 4th international symposium on Neural Networks: Part II–Advances in Neural Networks*, ISNN '07, pages 1221–1230, Berlin, Heidelberg. Springer-Verlag.

Bernal, A., Crammer, K., Hatzigeorgiou, A., and Pereira, F. (2007). Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol*, 3(3):e54.

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.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267.

Cai, D., Delcher, A., Kao, B., and Kasif, S. (2000). Modeling splice sites with Bayes networks. *Bioinformatics*, 16(2):152–158.

Catal, C. and Diri, B. (2009). Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction. *Expert Systems*, 26(5):458–471.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6.

Chelba, C. and Acero, A. (2006). Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naïve Bayes classifiers for text classification. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 540. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Emanuelsson, O. (2002). Predicting protein subcellular localisation from amino acid sequence information. *Briefings in bioinformatics*, 3(4):361–376.

Emanuelsson, O., Nielsen, H., Brunak, S., and von Heijne, G. (2000). Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of molecular biology*, 300(4):1005–1016.

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.

Gardy, J. L., Spencer, C., Wang, K., Ester, M., Tusnády, G. E., Simon, I., Hua, S., deFays, K., Lambert, C., Nakai, K., and Brinkman, F. S. (2003). PSORT-B: improving protein subcellular localization prediction for Gram-negative bacteria. *Nucleic Acids Research*, 31(13):3613–3617.

Giannoulis, G., Krithara, A., Karatsalos, C., and Paliouras, G. (2014). Splice site recognition using transfer learning. In *SETN*, pages 341–353. Springer.

Gross, S. S., Do, C. B., Sirota, M., and Batzoglou, S. (2007). Contrast: a discriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome biology*, 8(12):R269.

He, H. and Garcia, E. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Herndon, N. and Caragea, D. (2013). Naïve Bayes Domain Adaptation for Biological Sequences. In *Proceedings of the 4th International Conference on Bioinformatics Models, Methods and Algorithms*, BIOINFORMATICS 2013, pages 62–70.

Herndon, N. and Caragea, D. (2014a). Empirical Study of Domain Adaptation with Naïve Bayes on the Task of Splice Site Prediction. In *Proceedings of the 5th International Conference on Bioinformatics Models, Methods and Algorithms*, BIOINFORMATICS 2014, pages 57–67.

Herndon, N. and Caragea, D. (2014b). Predicting protein localization using a domain adaptation approach. In *Biomedical Engineering Systems and Technologies*, pages 191–206. Springer.

Herndon, N. and Caragea, D. (2015a). Domain adaptation with logistic regression for the task of splice site prediction. In *Bioinformatics Research and Applications*, pages 125–137. Springer.

Herndon, N. and Caragea, D. (2015b). Empirical study of domain adaptation algorithms on the task of splice site prediction. In *Biomedical Engineering Systems and Technologies*, pages 195–211. Springer.

Herndon, N. and Caragea, D. (2015c). An evaluation of self-training styles for domain adaptation on the task of splice site prediction. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1042–1047. ACM.

Herndon, N. and Caragea, D. (2016a). Ab initio splice site prediction with simple domain adaptation classifiers. In *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 3: BIOINFORMATICS*, pages 245–252.

Herndon, N. and Caragea, D. (2016b). A study of domain adaptation classifiers derived from logistic regression for the task of splice site prediction.

Herndon, N., Tangirala, K., and Caragea, D. (2014). Predicting protein localization using a domain adaptation naïve Bayes classifier with Burrows Wheeler transform features. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 501–504. IEEE.

Hubbard, T. and Park, J. (1995). Fold recognition and ab initio structure predictions using hidden Markov models and beta-strand pair potentials. *Proteins*, 23(3):398–402.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.

Korf, I., Flicek, P., Duan, D., and Brent, M. R. (2001). Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17(suppl 1):S140–S148.

Le Cessie, S. and Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Applied statistics*, pages 191–201.

Li, H., Guevara, N., Herndon, N., Caragea, D., Neppalli, K., Caragea, C., Squicciarini, A., and Tapia, A. H. (2015). Twitter Mining for Disaster Response: A Domain Adaptation Approach. In *Proceedings of the 12th International Conference on Information Systems for Crisis Response and Management*, ISCRAM 2015.

Li, J., Wang, L., Wang, H., Bai, L., and Yuan, Z. (2012). High-accuracy splice site prediction based on sequence component and position features. *Genet Mol Res*, 11(3):3432–51.

Li, Y., Li-Byarlay, H., Burns, P., Borodovsky, M., Robinson, G. E., and Ma, J. (2013). Truesight: a new algorithm for splice junction detection using rna-seq. *Nucleic acids research*, 41(4):e51–e51.

Maeireizo, B., Litman, D., and Hwa, R. (2004). Co-training for predicting emotions with spoken dialogue data. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naïve Bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Minoche, A. E., Dohm, J. C., Schneider, J., Holtgräwe, D., Viehöver, P., Montfort, M., Sörensen, T. R., Weisshaar, B., and Himmelbauer, H. (2015). Exploiting single-molecule transcript sequencing for eukaryotic gene prediction. *Genome biology*, 16(1):1–13.

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, S., and Schölkopf, B. (2001). An Introduction to Kernel-Based learning Algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202.

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: a comparison of logistic regression and naïve bayes. In *Proceedings of the Neural Information Processing Systems Conference*, pages 841–848.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Noble, W. S. (2006). What is a support vector machine? *Nat Biotech*, 24(12):1565–1567.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Raithel, S., Johnson, L., Galliart, M., Brown, S., Shelton, J., Herndon, N., and Bello, N. M. (2016). Inferential considerations for low-count RNA-seq transcripts: a case study on the dominant prairie grass andropogon gerardii. *BMC genomics*, 17(1):1.

Rätsch, G., Sonnenburg, S., Srinivasan, J., Witte, H., Müller, K.-R., Sommer, R., and Schölkopf, B. (2007). Improving the C. elegans genome annotation using machine learning. *PLoS Computational Biology*, 3:e20.

Riloff, E., Wiebe, J., and Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roli, F. and Marcialis, G. (2006). Semi-supervised pca-based face recognition using self-training. In Yeung, D.-Y., Kwok, J., Fred, A., Roli, F., and de Ridder, D., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 560–568. Springer Berlin Heidelberg.

Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., Ranganath, V. P., Li, H., and Guevara, N. (2015). Experimental study with real-world data for android app security analysis using machine learning. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 81–90. ACM.

Schweikert, G., Widmer, C., Schölkopf, B., and Rätsch, G. (2008). An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In *NIPS'08*, pages 1433–1440.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

Shelton, J. M., Coleman, M. C., Herndon, N., Lu, N., Lam, E. T., Anantharaman, T., Sheth, P., and Brown, S. J. (2015). Tools and pipelines for bionano data: molecule assembly pipeline and fasta super scaffolding tool. *BMC genomics*, 16(1):1.

Sonnenburg, S., Schweikert, G., Philips, P., Behr, J., and Rätsch, G. (2007). Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(Supplement 10):1–16.

Stanescu, A. and Caragea, D. (2014a). Ensemble-based semi-supervised learning approaches for imbalanced splice site datasets. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 432–437. IEEE.

Stanescu, A. and Caragea, D. (2014b). Semi-supervised self-training approaches for imbalanced splice site datasets. In *Proceedings of the 6th International Conference on Bioinformatics and Computational Biology, BICoB*, pages 131–136.

Stanke, M. and Waack, S. (2003). Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19(suppl 2):ii215–ii225.

Steijger, T., Abril, J. F., Engström, P. G., Kokocinski, F., Hubbard, T. J., Guigó, R., Harrow, J., Bertone, P., Consortium, R., et al. (2013). Assessment of transcript reconstruction methods for RNA-seq. *Nature methods*, 10(12):1177–1184.

Tan, S., Cheng, X., Wang, Y., and Xu, H. (2009). Adapting naïve Bayes to domain adaptation for sentiment analysis. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 337–349, Berlin, Heidelberg. Springer-Verlag.

Tangirala, K., Herndon, N., and Caragea, D. (2015). Community detection-based feature construction for protein sequence classification. In *Bioinformatics Research and Applications*, pages 331–342. Springer.

Tangirala, K., Herndon, N., and Caragea, D. (2016). A comparative analysis between k-mers and community detection-based features for the task of protein classification.

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.

Zhang, Y., Chu, C.-H., Chen, Y., Zha, H., and Ji, X. (2006). Splice site prediction using support vector machines with a Bayes kernel. *Expert Syst. Appl.*, 30(1):73–81.

Zhao, C., Escalante, L. N., Chen, H., Benatti, T. R., Qu, J., Chellapilla, S., Waterhouse, R. M., Wheeler, D., Andersson, M. N., Bao, R., et al. (2015). A massive expansion of effector genes underlies gall-formation in the wheat pest mayetiola destructor. *Current Biology*, 25(5):613–620.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer.

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., and Müller, K.-R. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807.