Published in final edited form as:

IEEE Trans Nanobioscience. 2016 March ; 15(2): 75–83. doi:10.1109/TNB.2016.2522400.

A Study of Domain Adaptation Classifiers Derived from Logistic Regression for the Task of Splice Site Prediction

Nic Herndon and Doina Caragea [Member, IEEE]

Department of Computing and Information Sciences, Kansas State University, Manhattan, Kansas 66506, United States

Nic Herndon: nherndon@ksu.edu; Doina Caragea: dcaragea@ksu.edu

Abstract

Supervised classifiers are highly dependent on abundant labeled training data. Alternatives for addressing the lack of labeled data include: labeling data (but this is costly and time consuming); training classifiers with abundant data from another domain (however, the classification accuracy usually decreases as the distance between domains increases); or complementing the limited labeled data with abundant unlabeled data from the same domain and learning semi-supervised classifiers (but the unlabeled data can mislead the classifier). A better alternative is to use both the abundant labeled data from a source domain, the limited labeled data and optionally the unlabeled data from the target domain to train classifiers in a domain adaptation setting. We propose two such classifiers, based on logistic regression, and evaluate them for the task of splice site prediction – a difficult and essential step in gene prediction. Our classifiers achieved high accuracy, with highest areas under the precision-recall curve between 50.83% and 82.61%.

Index Terms

Domain adaptation; logistic regression; splice site prediction; imbalanced data

I. Introduction

THE adoption of next generation sequencing (NGS) technologies a few years ago has led to both opportunities and challenges. The NGS technologies have made it affordable to sequence new organisms but have also produced a large volume of data that need to be organized, analyzed, and interpreted to create or improve, for example, genome assemblies or genome annotations. For genome annotation a major task is to accurately identify the splice sites – the regions of DNA that separate the exons from introns (donor splice sites), and the introns from exons (acceptor splice sites). The majority of the donor and acceptor splice sites, also known as canonical sites, are the GT and AG dimers, respectively, but not all GT, AG dimers are splice sites. Only about 1% or less of them are [1], making the splice site prediction a difficult task.

NGS technologies have also enabled better gene predictions through programs that assemble short RNA-Seq reads into transcripts and then align them on the genome. For example,

TWINSKAN [2] and CONTRAST [3] model the entire transcript structure as well as conservation in related species. Still, transcript assemblies from RNA-Seq reads are not error proof, and should be subjected to independent validation [4].

Supervised machine learning algorithms, which have been successfully applied on many biological problems, including gene prediction, could be seen as alternative tools for such validation. For example, support vector machines (SVM) have been used for identification of translation initiation sites [5], [6], labeling gene expression profiles as malign or benign [7], *ab initio* gene prediction [8], and protein function prediction [9], whereas hidden Markov models have been used for *ab initio* gene predictions [10], [11], among others.

However, supervised machine learning algorithms require large amounts of labeled data to learn accurate classifiers. Yet, for many biological problems, including splice site prediction, labeled data may not be available for an organism of interest. An option would be to label enough data from the target domain for a supervised target classifier, but this is time consuming and costly. Another option is to complement the limited labeled data with abundant unlabeled data from the same target domain and learn semi-supervised classifiers. However, it can happen that a classifier is degraded by the unlabeled data [12]. 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), another option is to learn a classifier from the related organism. 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.

A better alternative is to learn a classifier in a domain adaptation framework. In this setting, the large corpus of labeled data from the related, well studied organism is used in conjunction with available labeled data from the new organism, and optionally any unlabeled data from the new organism, to produce an accurate classifier for the latter. There are challenges with this approach, 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 the target unlabeled data, as adding unlabeled data could decrease the accuracy of the classifier. In addition, if the labeled data is used, how should it be added: iteratively or all at once, with hard labels, soft labels, or a combination of both?

In this work, we propose two domain adaptation approaches, presented in Sect. III-C and Sect. III-D, based on the supervised logistic regression classifier described in Sect. III-A. These approaches are simple, yet highly accurate. When trained on a source organism, *C.elegans*, and one of four target organisms, *C.remanei*, *P.pacificus*, *D.melanogaster*, and *A.thaliana*, with data described in Sect. III-E, these algorithms achieved high accuracy, with

highest areas under the precision-recall curve between 50.83% for distant domains and 82.61% for closely related domains, as shown in Sect. IV.

II. Related Work

Most of the approaches addressing splice site prediction involve supervised learning. For example, Li *et al.* [13] 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.* [14], Sonnenburg *et al.* [1], and Zhang *et al.* [15], also proposed supervised support vector machine classifiers, whereas Baten *et al.* [16] proposed a method using a hidden Markov model, Cai *et al.* [17] proposed a Bayesian network algorithm, and Arita, Tsuda, and Asai [18] 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.* [19]. However, one major drawback of these supervised algorithms is that they typically require large amounts of labeled data to train a classifier.

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. However, semi-supervised classifiers could be misled by the unlabeled data, especially when there is hardly any labeled data [12]. 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. Stanescu and Caragea [20] 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 [21], 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%.

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, such as features' independence. For instance, in text classification, Dai *et al.* [22] 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 proposed by Tan *et al.* [23], identified and used only the generalizable features from the source domain, in conjunction with unlabeled data from the target domain. It produced promising results for several target domains when evaluated on the task of sentiment analysis.

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. For example, Herndon and Caragea [24] modified the algorithm proposed by Tan *et al.* [23], by using a small amount of labeled data from the target domain and incorporating self-training. Although this modified algorithm produced promising results on the task of protein localization, it performed poorly on the splice site prediction data. An improved version of that algorithm, [25], implemented further changes, such as normalizing the counts for the prior and likelihood, using mutual information in selecting generalizable features, and representing the DNA sequences with location aware features. With these changes, the method produced promising results on the task of splice site prediction, with values for the highest area under the precision-recall curve between 43.20% for distant domains and 78.01% for related domains. In a recent approach for splice site prediction, Giannoulis *et al.* [26], 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. Whereas this method seems promising, in its current version, it was less accurate than the method in [25] – with the best values for the area under receiver operating characteristic (auROC) curve below 70%. We did not include their results in our comparison, as although they used the same datasets, they reported the performance of their classifier using auROC values, whereas we used auPRC – a more appropriate metric for this highly imbalanced problem.

Some of 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 support vector machine classifier proposed by Schweikert *et al.* [27] (which used a weighted degree kernel proposed by Rätsch *et al.* [28]). Another classifier that obtained best results for the task of splice site prediction, especially when the source and target domains are closely related, or when there is quite a bit of labeled data for the target domain, is our method proposed in [29]. This classifier used a convex combination of a supervised logistic regression classifier trained on the source data and a supervised logistic regression classifier trained on the target data to approximate the posterior probability for the target domain. Both of these classifiers though do not utilize the abundant unlabeled data from the target domain.

III. Methods and Materials

In this section, we present the classifiers that we use in our experiments. The first three use only labeled data, whereas the fourth one uses both labeled and unlabeled data. We describe them in the context of a binary classification task since splice site prediction is a binary classification problem. The first classifier, proposed by Le Cessie and Van Houwelingen [30], is a supervised logistic regression classifier. We will use this as a baseline for our domain adaptation classifiers. The second classifier uses a method proposed by Chelba and Acero [31] for maximum entropy models. This is a logistic regression classifier for the domain adaptation setting. The third classifier is our first proposed classifier for the domain adaptation setting, that uses a convex combination of logistic regression classifiers trained on source and target data. The fourth classifier is the second domain adaptation classifier we propose, that leverages labeled data from the source domain, and labeled and unlabeled data from the target domain.

A. Logistic Regression with Regularized Parameters

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

$$p(y|x;\theta) = \begin{cases} g(\theta^T x) & , \text{ if } y=1 \\ 1 - g(\theta^T x) & , \text{ if } y=0 \end{cases} = [g(\theta^T x)]^y \cdot [1 - g(\theta^T x)]^{1-y}$$

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

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

$$l(\theta) = \log \prod_{i=1}^m p(y_i|x_i;\theta) = \log \prod_{i=1}^m [g(\theta^T x_i)]^{y_i} \cdot [1 - g(\theta^T x_i)]^{1-y_i} = \sum_{i=1}^m [y_i \log g(\theta^T x_i) + (1-y_i) \log (1 - g(\theta^T x_i))]$$

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

$$\theta = \arg \max_{\theta} [l(\theta) - \lambda \|\theta\|^2] \quad (1)$$

Note that x_i is the i^{th} row in X , in our case, the i^{th} DNA sequence in the training data set, y_i is the i^{th} element of y , i.e., the corresponding label of x_i , and $x_0 = 1$, $\forall i \in \{1, 2, \dots, m\}$ such that $\theta^T x_i = \theta_0 + \sum_{j=1}^n \theta_j x_{ij}$.

B. Logistic Regression for Domain Adaptation Setting with Modified Regularization Term

The method proposed in [31] 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).

$$\theta_S = \arg \max_{\theta_S} [l(\theta_S) - \lambda \|\theta_S\|^2] \quad (2)$$

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:

$$\theta_T = \arg \max_{\theta_T} [l(\theta_T) - \lambda \|\theta_T - \theta_S\|^2] \quad (3)$$

C. Logistic Regression for Domain Adaptation Setting with Convex Combination of Posterior Probabilities

The first method we are proposing uses a convex combination of two logistic regression classifiers – one trained on the source data, and the other trained on the target data. First, we learn a model for the source domain and a model for the target domain, using the training instances from the source domain, (X_S, y_S) and from the target domain, (X_T, y_T) , respectively:

$$\theta_S = \arg \max_{\theta_S} [l(\theta_S) - \lambda \|\theta_S\|^2] \quad (4)$$

$$\theta_T = \arg \max_{\theta_T} [l(\theta_T) - \lambda \|\theta_T\|^2] \quad (5)$$

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

$$p(y|x;\alpha) \propto (1 - \alpha) \cdot p_S(y|x;\theta_S) + \alpha \cdot p_T(y|x;\theta_T) \quad (6)$$

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.

D. Logistic Regression for Domain Adaptation Setting that Incorporates Target Unlabeled Data

The second method we are proposing is a modified version of the algorithm we proposed in [25]. We made a couple of changes to that algorithm. First, we used the logistic regression with regularized parameters as the supervised classifier, instead of the naïve Bayes classifier used in [25], and second, we evaluated three variants of incorporating the target unlabeled data, as described below. To make the results comparable to the ones of the algorithm proposed in [25], we made no additional changes (besides the two mentioned above) – i.e., we filter out features from the source domain, even though we have not done the same 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 III-C: 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, we filter out the domain specific features from the source domain, and keep only the top ranking features, ranked with $f(w_i)$, that have similar mutual information in both the source and target labeled data sets:

$$f(w_i) = \frac{I_{tSL}(w_i; c_k) \cdot I_{tTL}(w_i; c_k)}{|I_{tSL}(w_i; c_k) - I_{tTL}(w_i; c_k)| + \rho} \quad (7)$$

where $I(w_i; c_k)$ is the mutual information between feature w_i and class c_k 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 our experiments we set $\rho = 4.9E^{-324}$, for minimal influence on the value of $f(w_i)$.

Once we filter out the domain specific features from the source domain, we estimate the posterior probabilities from the training source labeled data, p_{tSL} , and from the training target labeled data, p_{tTL} , using the regularized logistic regression classifier. With these probabilities we then label the instances from the training target unlabeled dataset:

$$p = \alpha \cdot p_{tTL} + (1 - \alpha) \cdot p_{tSL} \quad (8)$$

where $\alpha \in (0, 1)$ is a parameter that determines how much weight we assigned to source and target instances. We use three variants to label the instances from training unlabeled dataset. In one variant, we assign soft labels to all instances. By *soft* labels we mean that if for an instance d_i , our classifier predicts, using Equation (8), that $p(y = 1 | x) = 0.8$ and $p(y = 0 | x) = 0.2$, then we label it with $y = (0.8, 0.2)$. In another variant, we assign hard labels 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 we hard-label 100 instances, 1 positive and 99 negative. By *hard* labels we mean that if for an instance x we have $p(y = 1 | x) = 0.8$ and $p(y = 0 | x) = 0.2$, then we label it with $y = (1, 0)$. In the other variant, at each iteration we label the most confident instances with hard labels, and the remaining ones with soft labels.

Then we incorporate the instances from the training target unlabeled dataset, build a new classifier, predict the labels for the soft-labeled instances, and loop until the labels assigned to unlabeled data no longer change:

$$p = \alpha \cdot p_{tTL} + (1 - \alpha) \cdot [\gamma \cdot p_{tTU} + (1 - \gamma) \cdot p_{tSL}] \quad (9)$$

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 we compare the hard labels assigned to target unlabeled instances (i.e., for the purpose of checking for convergence, we temporarily assign hard labels to all instances from the training target unlabeled dataset, which we use for comparison between iterations). The pseudo-code for this classifier is shown in Classifier 1.

E. Data Set

We evaluated our proposed algorithms on the splice site dataset¹ first introduced in [27]. This contains DNA sequences from five organisms, *C.elegans* used as the source domain and four other organisms at increasing evolutionary distance from it, *C.remanei*, *P.pacificus*, *D.melanogaster*, and *A.thaliana*, as target domains. Each instance is a 141 nucleotides long DNA sequence, with the AG dimer at the sixty-first position, along with a label that indicates whether this AG dimer is an acceptor splice site or not. In each file 1 % of the instances are positive, i.e., the AG dimer at 61st position is an acceptor splice site, with small variations (variance is 0.01), whereas the remaining instances are negative. The data from the target organisms is split into three folds (by the authors who published the data in [27]) to obtain unbiased estimates for the classifier performance. Similar to [27], for our experiments, we used the training set of 100,000 instances from *C.elegans*, and the three folds of 2,500, 6,500, 16,000, and 40,000 labeled instances, as well as the 100,000 instances from the other organisms as unlabeled data, and for testing, three folds of 20,000 instances each, from the target organisms. This allows us to compare our results with the previous state-of-the-art results on this dataset in [27]. Note that although the dataset we used only has acceptor splice sites, the problem of predicting donor splice sites can be addressed with the same approach.

F. Data Preparation and Experimental Setup

We use two similar representations for the data. In one of them, we convert each DNA sequence into a set of features that represent the nucleotides present in the sequence at each position, and the trimer at each position. For example, given a DNA sequence starting with AAGATTCGC ... and label -1 we represent it as A, A, G, A, T, T, C, G, C, ..., AAG, AGA, GAT, ..., -1.

Classifier 1 Domain adaptation with logistic regression incorporating target unlabeled data

- 1: Remove domain specific features from the source dataset, using Equation (7).
- 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 (8).
- 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**

¹Downloaded from <ftp://ftp.tuebingen.mpg.de/fml/cwidmer/>

- 6: **M-step:** Train a classifier using Equation (9), 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 (9), on new target instances.

With these features we create a compact representation of a balanced combination of simple features in each DNA sequence, i.e., the 1-mers, and more complex features – features that capture the correlation between the nucleotides, i.e., the 3-mers. However, when the training data has a small number of instances, the trimers lead to a set of sparse features which can result in decreased classification accuracy. Therefore, in the other representation we keep only the nucleotide features. For an example DNA sequence starting with AAGATTCGC ... and label -1 we represent it as A, A, G, A, T, T, C, G, C, ..., -1.

We use these representations for two reasons. First, with these representations we achieved good results in [25] with a naïve Bayes classifier in a domain adaptation setting. And second, this allows us to compare the results of our proposed method with our previous results.

To find the optimal parameters' values we used the three folds of 100,000 instances from the target domain as validation sets. We first did a grid search for λ , using the baseline, supervised logistic classifier, with $\lambda = 10^x$, $x \in \{-8, -6, \dots, 4\}$, trained with data from source and target domains. For these datasets we got the best results when $\lambda = 1,000$. Therefore, for our proposed algorithm we set λ_S and λ_T to 1,000, and did a grid search for δ with values from $\{0.1, 0.2, \dots, 0.9\}$, whereas for our implementation of the method proposed in [31] we set λ_S to 1,000 and did a grid search for λ_T with $\lambda_T = 10^x$, $x \in \{-8, -7, \dots, 4\}$. We tuned λ_T for the method in [31], as λ_T controls the trade-off between source and target parameters, and thus it is similar to the δ parameter for our first proposed method. For our second proposed method we did a grid search 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 we trained on source and target data, whereas for the baseline classifiers, the supervised logistic regression, in one setting we trained on source, and in another setting we trained on each of the labeled target data set sizes: 2,500, 6,500, 16,000, and 40,000. To evaluate the classifiers we tested them on the test target data from the corresponding fold. We expect the results of the baseline, logistic regression classifier trained on each of the target labeled data sets to be the lower bound for our proposed method trained on the source data and that corresponding target labeled data, since we believe that adding data from a related organism should produce a better classifier.

All results are reported as averages over three random train-test splits to ensure that our results are unbiased. To evaluate the classifiers we used the area under the precision-recall curve (auPRC) for the positive class, since the data are so highly imbalanced [32]. We provide an example in Figure 1 that shows that for imbalanced datasets auPRC is a more appropriate measure than auROC.

With this experimental setup we wanted to evaluate:

1. The influence of the following factors on the performance of the classifier:
 - a. The features used: nucleotides (and trimers).
 - b. The amount of target labeled data: from 2,500, 6,500, 16,000 to 40,000 instances.
 - c. The evolutionary distance between organisms.
 - d. The weight assigned to the target data.
 - e. 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.
2. The performance of the domain adaptation classifiers derived from the supervised logistic regression classifier (the method proposed in [31], and our proposed methods), compared to other domain adaptation classifiers for the task of splice site prediction, i.e., an SVM classifier [27] and a naïve Bayes classifier [25].

IV. Results and Discussion

Table I shows the auPRC values of the minority class when using our proposed domain adaptation with convex combination of logistic regression classifiers, our domain adaptation classifier that uses target unlabeled data with its three variants for assigning labels to target unlabeled data, and, for comparison, when using the supervised logistic regression classifiers (trained on source or target), the logistic regression for domain adaptation classifier proposed in [31], the naïve Bayes classifier for domain adaptation from our previous work [25], and the best overall SVM classifier for domain adaptation proposed in [27], $SVM_{S,T}$. Based on these results, we make the following observations:

1. Factors that influence the performance of the classifier:
 - a. **Features:** our proposed classifiers performed better with nucleotide and trimer 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, our algorithm performs better with nucleotide features when there is little target labeled data. This conforms with our previous results [25], and with our intuition (see Section III-F): since trimers generate a sparse set of features, they lead to decreased classification accuracy when there are a small number of target training instances.
 - b. **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.
 - c. **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*, as shown in Table I. We believe this is a major reason that helps explain the decreased auPRC values for all classifiers, for these organisms, respectively, i.e., in general $\text{auPRC for } C.remanei > P.pacificus > \dots > A.thaliana$.

- d. **Weight assigned to target data:** intuitively, for our first proposed method, we expect 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 our 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 our second proposed method, we expect 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 we expect best results with increasing values for α , which assign more weight to the target labeled data. For β we expect the best results for high β values, as after a few iterations there should be enough confidently labeled data in the target domain. The results confirm our intuition, as shown in Figure 2.
 - e. **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 trimers 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 we use 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 our 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 we hypothesize 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, we hypothesize that 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.
2. In terms of performance, the method proposed in [31] produced worse results than the supervised logistic regression classifier trained on the target data. We believe that these poor results are due to this method's modified optimization function, 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 our previous work [25], 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).

Our first proposed method produced better average results than the supervised logistic classifier trained on either the source or the target domain in *every* case of the 16 we evaluated. This confirms our 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 our previous work [25], as well as the method proposed in [31] in every case, and outperformed the best overall domain adaptation SVM classifier proposed in [27] in 9 out of the 16 cases.

Our second proposed method 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 we evaluated. It also outperformed the domain adaptation naïve Bayes classifier proposed in our previous work [25], as well as the method proposed in [31] in every case, and outperformed the best overall domain adaptation SVM classifier proposed in [27] in 7 out of the 16 cases.

Based on these results we would recommend using our first proposed method over the domain adaptation 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, our second method when there is little labeled data, and the SVM algorithm proposed in [27] 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).

V. Conclusions and Future Work

In this paper we proposed two domain adaptation algorithms derived from the supervised logistic regression classifier for the task of splice site prediction. Our first proposed method uses a convex combination of a supervised logistic regression classifier trained on the source data and a supervised logistic regression classifier trained on the target data to approximate the posterior probability for every instance from the test set of the target domain. Our second proposed method uses a convex combination of supervised logistic regression classifiers (trained with source labeled data, target labeled data and target unlabeled data) and three variants of assigning labels to unlabeled data. We compared our algorithms with the domain adaptation classifier derived from the supervised logistic regression classifier, proposed in [31], the supervised logistic regression (as baseline), the SVM classifier proposed in [27], and the naïve Bayes classifier proposed in [25].

We evaluated these classifiers on four target domains of increasing distance from the source domain. Whereas the method proposed in [31] performed worse in most cases than the

domain adaptation naïve Bayes classifier proposed in our previous work [25], our first proposed method outperformed the best overall domain adaptation SVM classifier [27] in 9 out of the 16 cases, and our second proposed method produced the best results in 7 out of the 16 cases. In addition, both of our proposed methods outperformed the domain adaptation classifier proposed in [31], and our previous proposed method, [25]. Our empirical evaluation of these classifiers also provided evidence that the task of splice site prediction becomes more difficult as the complexity of the organism increases.

In future work, we would like to explore ways to improve the accuracy of the classifier, even with these highly imbalanced data. For example, we would like to randomly split the negative instances to create smaller balanced data sets. Then, we would train an ensemble of classifiers with the method we proposed in this paper. Furthermore, we would like to evaluate the effectiveness of our proposed method on other problems that can be addressed in a domain adaptation framework, e.g. text classification problems, sentiment analysis.

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.

References

1. Sonnenburg S, Schweikert G, Philips P, Behr J, Rätsch G. Accurate splice site prediction using support vector machines. *BMC bioinformatics*. 2007; 8(Suppl 10):S7. [PubMed: 18269701]
2. Korf I, Flicek P, Duan D, Brent MR. Integrating genomic homology into gene structure prediction. *Bioinformatics*. 2001; 17(suppl 1):S140–S148. [PubMed: 11473003]
3. Gross SS, Do CB, Sirota M, Batzoglou S. Contrast: a discriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome biology*. 2007; 8(12):R269. [PubMed: 18096039]
4. Steijger T, Abril JF, Engström PG, Kokocinski F, Hubbard TJ, Guigó R, Harrow J, Bertone P, et al. R. Consortium. Assessment of transcript reconstruction methods for rna-seq. *Nature methods*. 2013; 10(12):1177–1184. [PubMed: 24185837]
5. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B. An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*. 2001; 12(2):181–201.
6. Zien A, Rätsch G, Mika S, Schölkopf B, Lengauer T, Müller KR. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*. 2000; 16(9):799–807. [PubMed: 11108702]
7. Noble WS. What is a support vectormachine? *Nature biotechnology*. 2006; 24(12):1565–1567.
8. Bernal A, Crammer K, Hatzigeorgiou A, Pereira F. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol*. 2007; 3(3):e54. [PubMed: 17367206]
9. Brown MP, Grundy WN, Lin D, Cristianini N, Sugnet CW, Furey TS, Ares M, Haussler D. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*. 2000; 97(1):262–267.
10. Hubbard TJ, Park J. Fold recognition and ab initio structure predictions using hidden markov models and β -strand pair potentials. *Proteins: Structure, Function, and Bioinformatics*. 1995; 23(3):398–402.
11. Stanke M, Waack S. Gene prediction with a hidden markov model and a new intron submodel. *Bioinformatics*. 2003; 19(suppl 2):ii215–ii225. [PubMed: 14534192]

12. Catal C, Diri B. Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction. *Expert Systems*. 2009; 26(5):458–471.
13. Li J, Wang L, Wang H, Bai L, Yuan Z. High-accuracy splice site prediction based on sequence component and position features. *Genet Mol Res*. 2012; 11(3):3431–3451.
14. Baten AK, Chang BC, Halgamuge SK, Li J. Splice site identification using probabilistic parameters and svm classification. *BMC bioinformatics*. 2006; 7(Suppl 5):S15. [PubMed: 17254299]
15. Zhang Y, Chu CH, Chen Y, Zha H, Ji X. Splice site prediction using support vectormachines with a bayes kernel. *Expert Systems with Applications*. 2006; 30(1):73–81.
16. Baten, AK.; Halgamuge, SK.; Chang, B.; Wickramarachchi, N. *Advances in Neural Networks–ISNN 2007*. Springer; 2007. Biological sequence data preprocessing for classification: A case study in splice site identification; p. 1221-1230.
17. Cai D, Delcher A, Kao B, Kasif S. Modeling splice sites with bayes networks. *Bioinformatics*. 2000; 16(2):152–158. [PubMed: 10842737]
18. Arita M, Tsuda K, Asai K. Modeling splicing sites with pairwise correlations. *Bioinformatics*. 2002; 18(suppl 2):S27–S34. [PubMed: 12385980]
19. Al-Turaiki, IM.; Mathkour, H.; Touir, A.; Hammami, S. *Informatics Engineering and Information Science*. Springer; 2011. Computational approaches for gene prediction: A comparative survey; p. 14-25.
20. Stanescu, A.; Caragea, D. *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*. IEEE; 2014. Ensemble-based semi-supervised learning approaches for imbalanced splice site datasets; p. 432-437.
21. Stanescu, A.; Caragea, D. Semi-supervised self-training approaches for imbalanced splice site datasets; *Proceedings of the 6th International Conference on Bioinformatics and Computational Biology, BICoB*; 2014. p. 131-136.
22. Dai, W.; Xue, GR.; Yang, Q.; Yu, Y. *Proceedings of the national conference on artificial intelligence*. Vol. 22. Menlo Park, CA; Cambridge, MA; London: AAAI Press; MIT Press; 2007. Transferring naive bayes classifiers for text classification; p. 5401999
23. Tan, S.; Cheng, X.; Wang, Y.; Xu, H. *Advances in Information Retrieval*. Springer; 2009. Adapting naive bayes to domain adaptation for sentiment analysis; p. 337-349.
24. Herndon, N.; Caragea, D. *Biomedical Engineering Systems and Technologies*. Springer; 2014. Predicting protein localization using a domain adaptation approach; p. 191-206.
25. Herndon N, Caragea D. Empirical study of domain adaptation with naïve bayes on the task of splice site prediction. *Proceedings of the 5th International Conference on Bioinformatics Models, Methods and Algorithms*, ser *BIOINFORMATICS*. 2014; 2014:57–67.
26. Giannoulis, G.; Krithara, A.; Karatsalos, C.; Paliouras, G. *SETN*. Springer; 2014. Splice site recognition using transfer learning; p. 341-353.
27. Schweikert G, Rätsch G, Widmer C, Schölkopf B. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. *Advances in Neural Information Processing Systems*. 2009:1433–1440.
28. Sonnenburg S, Schweikert G, Philips P, Behr J, Rätsch G. Accurate splice site prediction using support vector machines. *BMC bioinformatics*. 2007; 8(Suppl 10):S7. [PubMed: 18269701]
29. Herndon N, Caragea D. Domain adaptation with logistic regression for the task of splice site prediction. *11th International Symposium on Bioinformatics Research and Applications*, ser *ISBRA*. 2015; 2015:125–137.
30. Le Cessie S, Van Houwelingen JC. Ridge estimators in logistic regression. *Applied statistics*. 1992:191–201.
31. Chelba C, Acero A. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*. 2006; 20(4):382–399.
32. Davis, J.; Goadrich, M. *Proceedings of the 23rd international conference on Machine learning*. ACM; 2006. The relationship between precision-recall and roc curves; p. 233-240.

Fig. 1.

auROC and auPRC values for our first proposed method 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.

Fig. 2.

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 .

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_SL classifier is the baseline logistic regression classifier trained on 100,000 instances from the source domain, *C.elegans* (first and tenth rows), and target labeled data (second and eleventh rows). The LR_DA_{cc} and LR_DA_{reg} domain adaptation classifiers are trained on a combination of source labeled and target labeled data, whereas the NB_DA_{s+h} and LR_DA_{s+h} domain adaptation classifiers are trained on a combination of source labeled, target labeled, and target unlabeled data. LR_DA_s assigns only soft labels to instances from target unlabeled dataset, LR_DA_{s+h} assigns at each iteration hard labels to the most confident predictions and soft labels to the remaining instances, and LR_DA_h assigns at each iteration hard labels to the most confident predictions. We show for comparison with our classifiers the values for the best overall classifier in [27], SVM_{S,7} (listed in these subtables as SVM), the values for our implementation of the LR_DA_{reg} classifier proposed in [31], and the values for the best overall classifier in [25], A1, (listed in these subtables as NB_DA_{s+h}). Note that the SVM classifier used different features. The best average value for each target dataset size is shown in bold. We would like to highlight that our first proposed classifier always performed better than the baseline classifier, and performed better in 9 out of 16 cases than the SVM classifier – the best classifier out of the three domain adaptation classifiers used for comparison with our classifiers. We couldn't check if the differences between our classifier and the SVM classifier are statistically significant, as we did not have the performance results per-fold for the SVM classifier (only average performance values were available in [27]). We would also like to note a common trend for our second proposed classifier, namely that in most cases using a combination of hard and soft labels generates better results than using soft labels only, which is better than using hard labels only. In addition, our second proposed method generated best results in 7 out of the 16 cases.

Table I

(a) <i>C.remanei</i>					
Features	Classifier	2,500	6,500	16,000	40,000
nucleotides	LR_SL _s	77.63±1.37			
	LR_SL _T	31.07±8.72	54.20±3.97	65.73±2.76	72.93±1.70
	LR_DA _{cc}	77.64±1.39	77.75±1.25	77.88±1.42	78.10±1.15
	LR_DA _{reg}	16.30±7.70	40.87±3.26	49.07±0.93	58.37±2.63
	NB_DA _{s+h}	59.18±1.17	63.10±1.23	63.95±2.08	63.80±1.41
	LR_DA _s	77.63±1.11	77.76±0.98	77.86±1.10	78.02±0.85
	LR_DA _h	77.39±1.03	77.58±0.88	77.80±1.07	77.89±0.81
	LR_DA _{s+h}	77.65±1.19	77.74±0.92	77.87±1.16	78.04±0.85
	SVM	77.06±2.13	77.80±2.89	77.89±0.29	79.02±0.09
	LR_SL _s	81.37±2.27			
nucleotides and trimers					

(a) <i>C.remanei</i>					
Features	Classifier	2,500	6,500	16,000	40,000
	LR_SL _T	26.93±9.91	55.26±2.21	68.30±1.91	77.33±2.78
	LR_DA _{cc}	81.39±2.30	81.47 ±2.19	81.78 ±2.08	82.61 ±2.00
	LR_DA _{reg}	2.30±1.05	14.50±4.68	40.10±3.72	63.53±7.10
	NB_DA _{sp,h}	45.29±2.62	72.00±4.16	74.83±4.32	77.07±4.45
	LR_DA _s	81.05±1.82	81.04±1.39	67.95±1.53	76.97±2.26
	LR_DA _h	80.66±1.77	79.84±1.17	67.12±1.49	77.46±2.47
	LR_DA _{sp,h}	81.40 ±1.89	81.42±1.84	81.75±1.74	82.54±1.64
	(b) <i>P.pacificus</i>				
Features	Classifier	2,500	6,500	16,000	40,000
	LR_SL _S		64.20±1.91		
	LR_SL _T	29.87±3.58	49.03±4.90	59.93±2.74	69.10±2.25
	LR_DA _{cc}	64.70±1.85	65.31±2.10	66.76±0.89	70.18±2.12
	LR_DA _{reg}	18.00±3.83	32.73±2.69	40.73±4.30	55.73±1.62
nucleotides	NB_DA _{sp,h}	45.32±2.68	49.82±2.58	52.09±2.04	54.62±1.51
	LR_DA _s	66.11 ±1.50	66.36±1.60	67.32±0.72	70.19±1.70
	LR_DA _h	63.98±1.66	64.70±1.77	66.31±0.61	69.95±1.72
	LR_DA _{sp,h}	64.82±1.46	65.46±1.92	67.03±0.85	70.20±1.70
SVM		64.72±3.75	66.39 ±0.66	68.44±0.67	71.00±0.38
	LR_SL _S		62.37±0.84		
	LR_SL _T	28.40±4.49	49.67±2.83	62.97±3.32	74.60±2.85
	LR_DA _{cc}	64.18±1.10	65.49±1.84	69.76 ±2.08	75.82 ±2.00
	LR_DA _{reg}	4.37±1.76	14.50±4.86	38.23±6.54	63.70±5.28
nucleotides and trimers	NB_DA _{sp,h}	20.21±1.17	53.29±3.08	62.33±3.60	69.88±4.04
	LR_DA _s	64.47±1.23	65.40±1.51	62.66±2.57	74.09±2.39
	LR_DA _h	61.16±1.33	63.13±1.92	60.66±3.53	74.64±2.60

(a) <i>C.remanei</i>					
Features	Classifier	2,500	6,500	16,000	40,000
	LR_DA _{s+h}	64.55±1.05	65.59±1.68	68.71±1.29	74.81±1.62
(c) <i>D.melanogaster</i>					
Features	Classifier	2,500	6,500	16,000	40,000
	LR_SL _S		35.87±2.32		
	LR_SL _T	19.97±3.48	31.80±3.86	42.37±2.15	50.53±1.80
	LR_DA _{ce}	39.70±2.82	42.19±3.41	49.72±2.01	53.43±0.89
	LR_DA _{reg}	11.33±1.36	22.80±2.60	27.30±3.92	42.67±0.76
nucleotides					
	NB_DA _{s+h}	33.31±3.71	36.43±2.18	40.32±2.04	42.37±1.51
	LR_DA _s	42.61±1.62	44.44±1.93	49.80±1.59	53.63±0.80
	LR_DA _h	48.02 ±1.10	47.24 ±1.27	50.18±1.73	53.76±0.80
	LR_DA _{s+h}	41.70±2.01	44.15±2.00	49.76±1.61	53.64±0.79
	SVM	40.80±2.18	37.87±3.77	52.33 ±0.91	58.17 ±1.50
(d) <i>A.thaliana</i>					
	LR_SL _S		32.23±2.76		
	LR_SL _T	15.07±4.11	28.30±5.45	44.67±3.23	38.43±32.36
	LR_DA _{ce}	37.24±2.20	40.93±3.79	50.54±3.91	45.89±22.25
	LR_DA _{reg}	3.40±1.82	8.37±2.48	21.20±2.85	26.50±22.44
nucleotides and trimers					
	NB_DA _{s+h}	25.83±2.35	32.58±5.83	39.10±1.82	47.49±3.44
	LR_DA _s	37.00±2.02	40.51±3.05	48.46±1.35	47.11±13.69
	LR_DA _h	33.29±2.48	37.57±3.69	47.26±1.87	41.96±21.19
	LR_DA _{s+h}	37.15±2.03	40.80±3.03	50.82±2.70	48.35±15.26
(d) <i>A.thaliana</i>					
Features	Classifier	2,500	6,500	16,000	40,000
	LR_SL _S		16.93±0.21		
	LR_SL _T	13.87±2.63	26.03±3.29	38.43±6.18	49.33±4.07
	LR_DA _{ce}	20.67±0.58	27.19±1.30	40.56±3.26	49.75±2.82
	LR_DA _{reg}	8.50±2.08	17.93±4.72	23.30±2.35	39.10±4.97

(a) <i>C.remanei</i>					
Features	Classifier	2,500	6,500	16,000	40,000
	NB_DA _{qth}	18.46±1.13	25.04±0.72	31.47±3.56	36.95±3.39
	LR_DA _s	25.84±0.48	32.50 ±1.17	43.03 ±3.75	50.59±3.50
	LR_DA _h	29.87 ±0.73	29.54±1.27	41.30±4.15	50.24±3.68
	LR_DA _{sth}	23.43±0.28	32.18±1.28	42.65±3.74	50.61±3.52
	SVM	24.21±3.41	27.30±1.46	38.49±1.59	49.75±1.46
	LR_SL _S	14.07±0.31			
	LR_SL _T	8.87±1.84	21.10±4.45	38.53±8.08	49.77±2.77
	LR_DA _{cc}	16.42±1.20	26.44±2.49	41.35±6.49	50.83 ±2.28
nucleotides and trimers	LR_DA _{reg}	2.50±0.10	8.27±1.60	20.03±3.36	30.27±2.57
	NB_DA _{qth}	3.99±0.43	13.96±2.42	33.62±6.31	43.20±3.78
	LR_DA _s	16.50±0.68	27.00±2.30	40.86±4.58	49.67±2.36
	LR_DA _h	13.15±0.34	21.63±2.04	39.50±3.87	49.49±2.16
	LR_DA _{sth}	16.64±1.11	27.34±2.25	41.76±5.21	50.57±2.04