DISTANCE-BASED LEARNING APPROACHES FOR MULTIPLE INSTANCE LEARNING

by

Özgür Emre Sivrikaya

B.S., Industrial Engineering, Boğaziçi University, 2010

M.S., Industrial Engineering, Boğaziçi University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor, Assist. Prof. Mustafa Gökçe Baydoğan, for his invaluable guidance, encouragement and endless support throughout my study. This dissertation would not have been possible without his great mentorship, trust and patience.

I am extremely grateful to my thesis committee members, Prof. Necati Aras, Assist. Prof. İnci Meliha Baytaş and Prof. Taylan Cemgil for their valuable guidance and contributions. I am also thankful to Assoc. Prof. Mehmet Gönen and Assist. Prof. Emel Şeyma Küçükaşcı for their constructive comments in my thesis defense jury. I would like to extend my gratitude to Prof. Gürhan Kök and Assist. Prof. Hande Küçükaydın for their guidance and support.

Thanks should also go to Erdem Banak and Mert Yüksekgönül for their valuable collaboration. I am deeply indebted to my friends, Nevcan Uludaş, Mete Adıyaman, Merve Şahin, Çağatay Tezcan and Mehmet Sarıkaya for their encouragement, suggestions and help. My special thanks go to Davut Şimşek, Koray Parkın and Çağlar Ata. I could not have completed this work without their tremendous encouragement and understanding.

Finally, words cannot express my gratitude to my family, especially my parents and my sister, for their endless support and love. Their belief in me has kept my motivation high during my graduate studies. This endeavor would not have been possible without them.

ABSTRACT

DISTANCE-BASED LEARNING APPROACHES FOR MULTIPLE INSTANCE LEARNING

Multiple Instance Learning (MIL) is a weakly supervised approach that focuses on the labeling of a set of instances (i.e. bags) where the label information of individual instances is generally unknown. Many of the earlier MIL studies focus on certain assumptions regarding the relationship between the bag and instance labels and devise supervised learning approaches. With the ambiguity in instance labels, these studies fail to generalize to the MIL problems with complex structures. To avoid these problems, researchers focus on embedding instance-level information to learn bag representations. In this context, dissimilarity-based representations are known to generalize well. This thesis proposes a novel framework in which each bag is represented by its dissimilarities to the prototypes. The framework consists learning mechanisms that provide fast and competitive results compared to the existing distance-based approaches on extensive benchmark data sets. The first approach is a simple model that provides a prototype generator from a given MIL data set. We aim to find out prototypes in the feature space to map the collection of instances (i.e. bags) to a distance feature space and simultaneously learn a linear classifier for MIL. The second proposal is a tree-based ensemble learning strategy that avoids complex tuning processes and heavy computational costs without sacrificing accuracy. The framework is enriched with the integration of the methods, parameter selection strategy, and ensemble design. Furthermore, the proposed methods are extended to the regression domain, namely Multiple Instance Regression (MIR). MIR is a less commonly studied area where the bag labels are real valued data instead of classes. The experiments show that the performances of all proposals are better than the state-of-the art approaches in the literature.

ÖZET

ÇOKLU ÖRNEKLE ÖĞRENME PROBLEMLERİ İÇİN UZAKLIK TABANLI ÖĞRENME YAKLAŞIMLARI

Çoklu Örnekle Öğrenme (ÇÖÖ), bireysel örneklerin etiket bilgilerinin genellikle bilinmediği ancak örnek gruplarının (torba) etiket bilgilerinin bilindiği durumlarda, torbaların etiketlenmesini hedefleyen yarı denetimli bir öğrenme yaklaşımıdır. Mevcut ÇÖÖ çalışmalarının çoğu, torba ve örnek etiketleri arasındaki ilişkiyle ilgili belirli varsayımlara odaklanan denetimli öğrenme yaklaşımları sunmaktadır. Bu çalışmalar, örnek etiketlerinin belirsizliği nedeniyle, karmaşık yapılara sahip ÇÖÖ problemlerine genellenememektedir. Araştırmacılar, bu problemden kaçınmak için torba temsilini örnek seviyesindeki bilgiyi gömerek öğrenmeye odaklanmışlardır. Bu bağlamda, farklılığa dayalı temsillerin iyi genellenebildiği bilinmektedir. Bu tez, her bir torbanın prototiplere olan benzerlikleriyle temsil edildiği yeni bir yapı önermektedir. Bu yapı, kapsamlı kıyaslama veri setlerinde, mevcut benzerlik tabanlı yaklaşımlardan daha hızlı ve daha başarılı sonuçlar veren öğrenme yaklaşımlarından oluşmaktadır. Önerilen ilk yaklaşım, basit bir model ile ÇÖÖ veri setinden prototip oluşturmaktadır. Bu yaklaşımda, torbaları bir özellik uzayına eşlemek için özellik uzayında prototipler bulmayı ve aynı anda ÇÖÖ için doğrusal bir sınıflandırıcı öğrenme amaçlanmaktadır. İkinci öneri ise model performansından ödün vermeyerek, karmaşık değişken ayarlama süreçlerinden ve yüksek hesaplama maliyetlerinden kaçınan ağaç tabanlı bir topluluk öğrenme stratejisidir. Sunulan yapı, farklı yöntemlerin birleştirilmesi, değişken seçim stratejisi ve topluluk tasarımı ile zenginleştirilmiştir. Ayrıca, önerilen yöntemler regresyon alanına, yani Çoklu Örnekle Regresyon'a (ÇÖR) uygulanabilir hale getirilmiştir. ÇÖR, torba etiketlerinin sınıflar yerine gerçek değerli veriler olduğu ve literatürde daha az çalışılan bir alandır. Yapılan deneyler, bu tezde sunulan tüm önerilerin performanslarının literatürdeki bilinen yaklaşımlardan daha iyi olduğunu göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES xiii
LIST OF SYMBOLS xv
LIST OF ACRONYMS/ABBREVIATIONS xvii
1. INTRODUCTION 1
2. BACKGROUND 7
2.1. Definitions 7
2.2. Literature Review 8
2.3. Data Sets 15
3. PROTOTYPE LEARNING IN MULTIPLE INSTANCE LEARNING 20
3.1. Introduction 20
3.2. Distance Feature Extraction 21
3.3. Model Formulation 24
3.3.1. Feature Normalization 25
3.4. Interpretation 26
3.5. Time Complexity 27
3.6. Experimental Results 29
3.7. Classification Accuracy 30
3.8. Parameter Sensitivity 45
3.9. Conclusion 47
4. PROTOTYPE FOREST IN MULTIPLE INSTANCE LEARNING 48
4.1. Background 49
4.2. Prototype Forest 52
4.3. Prototype Forest with Prototype Learning 57
4.4. Linear Tree in Prototype Forest 60

4.5. Time Complexity 61
4.6. PCA Analysis of Data Sets 62
4.7. Experimental Results 68
4.8. Parameter Sensitivity 78
4.9. Time Complexity of the Algorithm 82
4.10. Conclusion 83
5. MULTIPLE INSTANCE REGRESSION 85
5.1. Model Formulation 85
5.2. Experimental Results 88
6. CONCLUSION AND FUTURE RESEARCH 94
REFERENCES 96
APPENDIX A: REGULARIZATION 108
APPENDIX B: ABOUT THE FIGURES USED IN THE DOCUMENT 110

LIST OF FIGURES

Figure 1.1. Bag to prototype dissimilarity definition. 4

Figure 3.1. Bag 1 and Bag 2 are negative labeled bags. Bag 3 and Bag 4 have positive labels. All bags and prototypes have two features. 21

Figure 3.2. Distances of bags to prototypes. A good prototype candidate separates negative and positive labeled bags. 22

Figure 3.3. Overview of PL architecture. The blue color shows the variables, which the model will be optimized over. L: Number of features in an instance, constant; D: Number of prototypes, hyperparameter; K_i: Number of instances bag i, varies between different bags . . . 22

Figure 3.4. MNIST image examples © 1998 IEEE [1]. 27

Figure 3.5. PL algorithm. 28

Figure 3.6. Prototypes of finding the 9 problem. Classifier coefficients of the prototypes: Left: 0.328, right: -0.359. 29

Figure 3.7. Prototypes of finding the 5 problem. Classifier coefficients of the prototypes: Left: 0.265, right: -0.359. 29

Figure 3.8. Mean AUC Performances. Numbers indicate the average ranks of the approach. 37

Figure 3.9. AUC comparison of PL and APR on benchmark data sets. 39

Figure 3.10. AUC comparison of PL and CCE on benchmark data sets. 39

Figure 3.11. AUC comparison of PL and Citation-KNN. 40

Figure 3.12. AUC comparison of PL and D_{maxmin} on benchmark data sets. . . 40

Figure 3.13. AUC comparison of PL and $D_{meanmin}$ on benchmark data sets. . . 41

Figure 3.14. AUC comparison of PL and D_{minmin} on benchmark data sets. . . . 41

Figure 3.15. AUC comparison of PL and MILES on benchmark data sets. . . . 42

Figure 3.16. AUC comparison of PL and miFV on benchmark data sets. 42

Figure 3.17. AUC comparison of PL and QP-MIL on benchmark data sets. . . 43

Figure 3.18. Average AUC performance under different learning rates of prototypes. 46

Figure 3.19. Average AUC performance under different learning rates of linear model weights. 46

Figure 3.20. Average AUC performance under different number of prototypes. . 47

Figure 4.1. Classification decision tree 1 for simple problem. 50

Figure 4.2. Classification decision tree 2 for simple problem. 51

Figure 4.3. Tree-based ensemble structure. 52

Figure 4.4. Overview of PF architecture. Bags are recursively partitioned based on their distances to randomly selected prototypes. 53

Figure 4.5. Entropy before and after the split. 55

Figure 4.6. Prototype tree algorithm. 56

Figure 4.7. Prototype tree trained on Musk 2. 57

Figure 4.8. TPL algorithm. 59

Figure 4.9. PFPL algorithm. 60

Figure 4.10. Linear model tree vs decision tree. 61

Figure 4.11. Linear tree in prototype tree algorithm. 62

Figure 4.12. Average distances between prototypes and bags when Web 8 instances are selected from bags of class 0. 63

Figure 4.13. Average distances between prototypes and bags when Web 8 instances are selected from bags of class 1. 63

Figure 4.14. Average distances between prototypes and bags when Hammonds Flycatcher instances are selected from bags of class 0. 64

Figure 4.15. Average distances between prototypes and bags when Hammonds Flycatcher instances are selected from bags of class 1. 64

Figure 4.16. Sample bags by bag id. 66

Figure 4.17. Sample bags by class. 66

Figure 4.18. Sample instance from Web 8. 67

Figure 4.19. Total variance explained by the first 20 principal components. . . 68

Figure 4.20. Explained variance by principal components for Web 8. 68

Figure 4.21. Mean AUC performances on benchmark data sets. Numbers indicate the average rank of the approach. 75

Figure 4.22. PFPL and LT pairwise comparison. 78

Figure 4.23. PFPL and PL pairwise comparison. 79

Figure 4.24. PF parameter sensitivity for D vs F. 80

Figure 4.25. PF parameter sensitivity for h vs τ. 80

Figure 4.26. PFPL parameter sensitivity for D vs F. 81

Figure 4.27. PFPL parameter sensitivity for h vs τ. 81

Figure 4.28. Average run time of PL under different problem parameters configurations. 82

Figure 4.29. Average run time of PFPL under different problem parameters configurations. 83

Figure 4.30. Average run time of PF under different h and τ configuration. . . 84

Figure 4.31. Average run time of PFPL under different h and τ configuration. . 84

Figure 5.1. PL-MIR algorithm. 87

Figure 5.2. PFPL-MIR algorithm. 89

Figure A.1. Musk2 prototypes with regularization. 108

Figure A.2. Musk2 prototypes without regularization. 109

LIST OF TABLES

Table 1.1. Labeling in MIL problems. 1

Table 1.2. Labeling in SIL problems. 2

Table 1.3. Representation of a simple MIL example. 2

Table 2.1. MIL data sets. 16

Table 3.1. Results of the algorithms. 31

Table 3.2. Results of the algorithms based on problem categories. 38

Table 3.3. Average run time of the algorithm (in seconds). 44

Table 4.1. A classification example. 50

Table 4.2. An extended classification example. 51

Table 4.3. MIL example. 54

Table 4.4. Bag to prototype distances. 54

Table 4.5. Mann-Whitney U test results. 65

Table 4.6. Results of the algorithms. 70

Table 4.7. Results of the algorithms based on problem categories. 77

Table 5.1. Results of the algorithms. 91

Table 5.2. Crop yield prediction results. 93

LIST OF SYMBOLS

$\|\cdot\|_1$	ℓ_1-norm
$\|\cdot\|_2$	ℓ_2-norm
B_i	Bag i
D	Number of prototypes
$Dist(\cdot,\cdot)$	Distance
E	Number of epochs
$E(\cdot)$	Entropy
F	Feature
h	Tree depth
K_i	Number of instances in bag i
L	Number of features
$\mathcal{L}_{ce}(\cdot,\cdot)$	Cross-entropy loss
M	Number of bags
$\max(\cdot)$	Maximum function
$\min(\cdot)$	Minimum function
n	Number of data set rows
N	Number of subset bags
p_i	Proportion of i in S
p^*	Split point
P_d	Prototype d
$P_{d,r}$	Feature r of prototype d
s	Element of feature matrix
S	Sample
X_{ij}	Instance j of bag i
$X_{ij,r}$	Feature r of instance j
y_i	Label of bag i
$\hat{y}_i$	Estimated label of bag i

α_w	Learning rates of classification weights
α_p	Learning rates of prototypes
β	Linear classifier weights
λ_d	Regularization parameter for the distance between the prototypes and the bags
λ_w	Regularization parameter for linear classifier weights
λ_p	Regularization parameter for prototypes
μ	Mean
$\mathcal{O}$	Time complexity
σ	Standard deviation
$\sigma(\cdot)$	Sigmoid function
τ	Number of trees
Φ_{id}^{min}	Minimum distance between bag i and prototype d
Φ_{id}^{max}	Maximum distance between bag i and prototype d
Φ_{id}^{mean}	Average distance between bag i and prototype d
$\hat{\chi}$	Subset of bags

LIST OF ACRONYMS/ABBREVIATIONS

3D	Three Dimensional
AOD	Aerosol Optical Depth
APR	Axis-parallel Rectangles
AUC	Area Under Receiver Operating Characteristic Curve
CCE	Constructive Clustering Ensemble
CD	Critical Difference
CPU	Central Processing Unit
CV	Cross Validation
DD	Diverse Density
EM-DD	Expectation Maximization Diverse Density
FPR	False Positive Rate
FV	Fisher Vector
GB	Gigabyte
GHz	Gigahertz
INFORMS	Institute for Operations Research and the Management Sciences
IG	Information Gain
KNN	K-Nearest Neighbors
LP	Linear Programming
LT	Linear Tree
miFV	Multi-Instance Learning based on Fisher Vector Representation
MIGraph	Multiple Instance Graph
MI	Multiple Instance
MIL	Multiple Instance Learning
MILES	Multiple Instance Learning via Embedded Instance Selection
MInD	Multiple Instance Dissimilarity
MIR	Multiple Instance Regression
MISVM	Multiple Instance Support Vector Machine

MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
NEURIPS	Conference on Neural Information Processing Systems
OOB	Out-of-Bag
PCA	Principal Component Analysis
PF	Prototype Forest
PFPL	Prototype Forest with Prototype Learning
PIR	Prime Instance Regression
PL	Prototype Learning
RF	Random Forest
RAM	Random Access Memory
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SE	Squared Error
SIL	Single Instance Learning
SVM	Support Vector Machine
TMA	Tissue Microarray
TPL	Tree-based Prototype Learning
TPR	True Positive Rate
UCSB	University of California, Santa Barbara
USDA	United States Department of Agriculture

1. INTRODUCTION

Multiple Instance Learning (MIL) is a very popular learning method that labels bag of instances data. In contrast to Single Instance Learning (SIL) problems, individual instances are not labeled in MIL. Instead, a collection of instances, namely a bag, has a label. Labeling characteristics of MIL and SIL problems can be found in Table 1.1 and Table 1.2 respectively. A common example of a MIL problem is from the computer vision domain. In image classification problems, images (i.e. bags) are represented by a set of feature vectors (i.e. instances) derived from the patches or key points of the image. In these problems, the label of the images are known but the instances do not have individual labels. MIL approaches aim at using the instance-level information to detect the bag's label [2] under this setting. This framework is also suitable for many other learning problems from text classification [3] to histopathology [4].

Table 1.1. Labeling in MIL problems.

Bag	Instance	Label
B_1	X_{11} X_{12}	1
B_2	X_{21} X_{22} X_{23}	0

A traditional MIL problem consist of some bags and each bag has a number of instances. A simple example of MIL representations can be found in Table 1.3. In this example, there are two bags B_1 and B_2, and each instance has two features. Number of instances in each bag may vary in MIL problems. In this thesis, instances are represented with X. B_1 has two instances X_{11} and X_{12}; B_2 has three instances X_{21}, X_{22} and X_{23}. Previous studies generally use this representation of bags and instances.

Table 1.2. Labeling in SIL problems.

Instance	Label
X_1	1
X_2	0
X_3	1
X_4	1
X_5	0

Table 1.3. Representation of a simple MIL example.

Bag	Label	Feature 1	Feature 2
B_1	1	$X_{11,1}$	$X_{11,2}$
		$X_{12,1}$	$X_{12,2}$
B_2	0	$X_{21,1}$	$X_{21,2}$
		$X_{22,1}$	$X_{22,2}$
		$X_{23,1}$	$X_{23,2}$

Solution approaches to MIL problems can be grouped into two main categories. The first one focuses on learning instance-level classifiers [5, 6]. The main advantage of instance-level approaches is the use of granular information. The main drawback is the unavailability of instance-level labels in MIL problems. Therefore, such approaches inherit bag labels in general. However, instances are not required to have bag's characteristics in MIL domain. As a result, the model may perform poorly with inherited bag labels. Another problem is that the granular information has many dimensions that lead to dimensionality problems in the solutions.

The second group is bag-level approaches [7, 8]. These approaches do not have dimensionality problems and solve the aforementioned performance problems. However, bag-level information is very high-level, and there is a potential to lose valuable

instance-level information. Both approaches have common characteristics in terms of learning a classifier with limited information [9]. The main difference between these two approaches is the unavailability of instance labels in MIL problems. However, the bag level classifiers can be used in instance level classification with some adjustments [10].

Embedding approaches are developed to overcome the problems of instance and bag-level approaches. These approaches embed instance or bag-level features into alternative feature space. A classifier is learned with the aggregation of features from the new space [2, 11]. Many state-of-the-art models, including the benchmark models considered in this thesis, adopt embedding approaches due to their performance superiority. However, embedding approaches, like instance-level and bag-level approaches, face certain challenges including computational cost of the algorithm (e.g. dimensionality), efficient parameter tuning, and loss of information (i.e. poor prediction performance). These challenges are further discussed in Section 2.2.

Existing MIL literature mainly focuses on the classification problems. However, instead of prediction of discrete labels, real values may also be desired to be predicted. Crop yield estimation and remote sensing data for aerosol retrieval are great examples for this need. The MIL problems for the real values are called Multiple Instance Regression (MIR) problems in the literature. There are only a couple of studies concentrates on the regression setting in MIL. The initial studies assume that there is a prime instance in each bag that is responsible for the bag's label [12]. However, this assumption potentially eliminates the important information that other instances carry. Therefore, to develop a powerful method for the common MIR problems, this assumption should be extended. As an extension to prime instance approach other approaches including prime instance clusters [13] and combination of instances [14] are proposed. Another method for MIR is adopting embedding approaches. These approaches concentrates on mapping the instance level information to another embedding space to utilize the most granular information [15, 16].

To avoid the aforementioned problems with the existing approaches, this study proposes new distance-based learning frameworks for MIL and MIR problems. Each bag is represented by its distances, namely dissimilarities, to reference objects, called prototypes. As a result, both instance level information and the computational efficiency of bag level approaches are preserved which provides the best results on the benchmark MIL data sets [11]. A simple illustration of bag to prototype dissimilarity features can be found in Figure 1.1. Let Figure 1.1 represents a two dimensional space and there is a bag, Bag i, and a prototype, Prototype d. Each dot in Bag i represents an instance. Bag i can be represented its minimum, maximum, and mean distance to Prototype d. Instance 1 has the minimum distance, solid line, to Prototype d. Instance 2 has the maximum distance, dashed line, to Prototype d. The average distance is the average distances of all instances of Bag i to Prototype d.

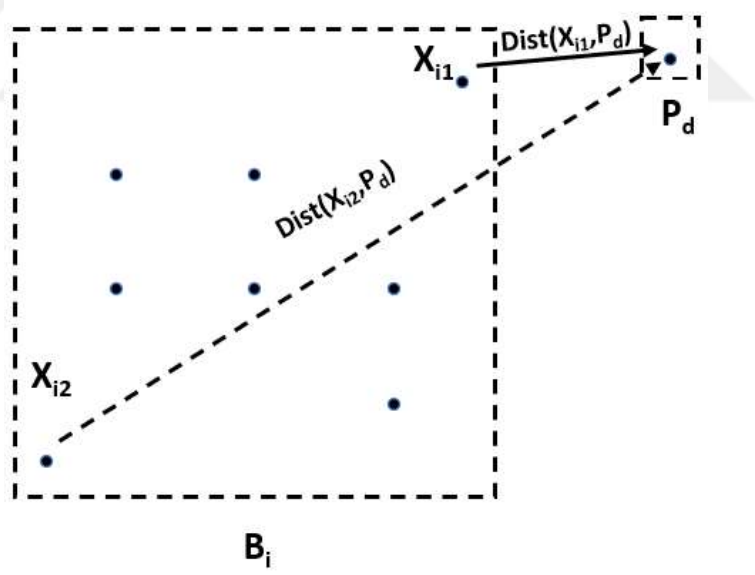


Figure 1.1. Bag to prototype dissimilarity definition.

The first proposed approach in this study is a simple model that provides a prototype generator from a given MIL data set. The study aims to find out prototypes in the feature space to map the bags to a distance feature space and simultaneously learn a linear classifier for MIL. The method is called Prototype Learning (PL). PL is published in Turkish Journal of Electrical Engineering and Computer Sciences [17]. The study is also presented in Informs 2019 Annual Meeting, Seattle and accepted to

NeurIPS 2019 Workshop, Sets and Partitions Workshop, Vancouver.

The first approach has many parameters to be tuned. Therefore, we enrich our proposal with another domain that can be executed faster. The second approach is a tree-based ensemble that constructs decision trees on the dissimilarity of the bags to randomly selected prototypes (i.e. bag-to-prototype distances) at each splitting step in decision tree learning. Due to the randomness associated with the prototype selection, a tree-based ensemble learning framework is devised to classify the bags with the proposed bag-to-prototype distance representation. The method is called Prototype Forest (PF).

The third proposal embeds PL into the PF. The motivation of the PL integration is to learn the prototypes that split the bags better than random prototypes. This work proposes three variations of PL and PF integration. In the first variation, a single decision tree is constructed with hyperparameter tuning which is called Tree-based Prototype Learning (TPL). In the second variation, PL hyperparameters are randomly selected among a parameter set which is called Prototype Forest with Prototype Learning (PFPL). The third variation splits the decision trees considering the logistic regression output instead of a constant approximation that traditional decision trees follow. The approach is called Linear Tree (LT). The details, advantages and disadvantages of all three proposals will be provided in the related chapters.

Finally, the proposed framework is modified for regression problems. The aforementioned proposals provide a solution to classification problems. However, in the multiple instance setting real values are also needed to be predicted. Our architecture can be applied to the regression domain with minor modifications (e.g. loss function). Therefore, we extend our proposal to the regression domain by changing the loss functions and split criteria of the decision trees.

In this thesis, we analyze the methods in multiple instance learning domain for classification and regression problems. We propose solution methods considering the

limitations of the existing literature. In this thesis, we investigate the answers to the following questions:

- MIL deals with the bag labels. However, the instances have important information on these labels. The instance level information has the dimensionality problem. How can we utilize the instance level information without sacrificing the fast run time? (Chapter 2)
- Prototypes are the feature vectors that may be utilized to estimate the bag's label. Are there efficient methods to learn prototypes or select them among the instances? (Chapter 3)
- Hyperparameter tuning is a time consuming process. How can we develop a learning method that does not have many hyperparameters to be tuned? Do tree-based ensemble methods propose an efficient learning procedure for the multiple instance domain? (Chapter 4)
- MIL setting has an extensive benchmark data sets varies from molecular activity prediction to image classification. Are there robust methods in MIL that outperforms other state-of-the-art methods for all data sets? (Chapter 3 and Chapter 4)
- Real values are also needed to be predicted in MIL domain. Are there any powerful methods to apply regression problems in MIL? (Chapter 5)

The following chapters provide the details of the study under the following structure. Chapter 2 gives a review of the domain and summarizes previous studies. Chapter 3 and Chapter 4 details the learning methods. Chapter 5 introduces MIR extension of the learning methods. Finally, Chapter 6 provides comments and summaries of the thesis.

2. BACKGROUND

2.1. Definitions

A bag i consists of a group instances and each instance is a L-dimensional feature vector. Let X_{ij} be the instance j of bag i and $X_{ij,r}$ be the rth feature of instance j of bag i. Bag i has a label of y_i and has K_i number of instances.

Let prototype d is defined by P_d. Prototype d is also represented by L features. Feature r of prototype d is defined as $P_{d,r}$. In the presented approach, distance features are extracted based on the bag to prototype dissimilarities. Pairwise instance to prototype distances are pooled to calculate bag to prototype dissimilarities. To explain the approach, Euclidean distance will be used for definitions. Euclidean distance of prototype d to instance X_{ij} is defined for each instance to prototype pair as

$$Dist(X_{ij}, P_d) = \sqrt{\sum_{r=1}^{L}(X_{ij,r} - P_{d,r})^2}. \tag{2.1}$$

Three alternative bag to prototype dissimilarity features are calculated using instance to prototype distances. These features are minimum (F_{min}), maximum (F_{max}) and mean (F_{mean}) which are defined as

$$F_{min}(B_i, P_d) = \min_j Dist(X_{ij}, P_d) \tag{2.2}$$

$$F_{max}(B_i, P_d) = \max_j Dist(X_{ij}, P_d) \tag{2.3}$$

$$F_{mean}(B_i, P_d) = \frac{1}{K_i}\sum_{j=1}^{K_i} Dist(X_{ij}, P_d) \tag{2.4}$$

respectively. These well-known features are useful for the generalization of the approach. Furthermore, certain data sets are affected by the noise due to instance characteristics. Using all three features prevents the algorithm from noise.

2.2. Literature Review

MIL problems have different categorizations based on instance or bag characteristics. We refer to [18] which divides MIL problems into three categories. These categories are determined by the instance characteristics. If all instances carry the bag's information, i.e. a bag of pictures of similar animal images, instances can be labeled with the bag's label. In the second category, one or more instances determine the bag's label, i.e. MIL problems with standard assumption. Standard MIL assumption tells that if a bag has at least one positive instance, then the bag's label is positive [19]. Some of the most famous MIL problems are designed with standard MIL assumption. Musk [19], a famous molecular activity prediction problem, is one of such problems. The problem's objective is the identification of musky molecules and the main assumption is that the musky smell comes from at least one positive conformation of a molecule. In this problem, each molecule is assumed to be a different bag. Conformations of the molecule are the instances. Several other approaches [20,21] adopt standard MIL assumption. An important note regarding the standard assumption is that this assumption is asymmetrical [22]. Therefore, the selection of positive and negative classes plays an important role in this assumption.

However, the standard MIL assumption should be extended for some other MIL problems. For example, if the objective of the problem is the identification of certain objects in a picture, a certain combination of instances is needed rather than a single positive instance. In other words, each instance adds to the image's label. This second assumption is called collective assumption [22, 23]. Classification of desert, beach and sea images is a good example of that requirement [22]. Let each image be a bag and each patch of an image be an instance. Identification of a single blue instance is not enough to label the image as either sea or beach. In this case, one instance is not enough for identification. Multiple blue instances without any sand instances are needed for labeling the picture as the sea. In such cases, instances (patches) do not have labels but the image has a label.

According to [18], in the third category, all instances carry some information about the bag's label. This case occurs when an instance carries only a portion of the bag's information and each instance contributes the bag's label. For example, there is a set of images that belong to the patches of a cat. In that case to label the bag as cat, all information that comes from all individual images is needed to be used. Even though certain patches may carry strong information on the image's label, more than one patch is needed for classification in general.

Even if the data set is in MIL domain, problems can be solved using non-MIL techniques. Especially, if each instance keeps all information of a bag label, instance labels may be used directly. Classification can be made using these labels. For example, consider different images of people and the main objective is to find the image of a specific person. This case can be formulated as an MIL problem by grouping several images and labeling the bag as positive if a bag includes that person's image. Each image, i.e. instance in MIL domain, is a pose from a different perspective and lighting. Therefore, it plays an important role in the bag's label. However, individual images can also be labeled and the problem can be reformulated as a non-MIL problem. In other words, a prediction model can be learned using instances and working in MIL domain is not required [24, 25].

Standard MIL assumption tells that if a bag has at least one positive instance, then the bag's label is positive [26]. Numerous research can be found using the standard assumption of MIL [19] and its variations and extensions such as the negative bags may have positive instances [27] and the standard assumption of MIL problem under the assumption that a certain amount of positive instances should be in a bag to label a bag positive [7, 28, 29]. The following example can be given for this category: There is a set of pictures from a concert and a learning method is needed to identify the lead vocal. In this case, it is known that the lead vocal is in one of the images. However, the exact image is not known. This is a regular MIL problem in which sets of images are bags and images are instances. The aim is to find the instance that makes the bag positive. In other words, the image that belongs to the lead vocal.

In such problems, bag level decision is made with the use of instance level information [30,31]. This approach is named as collective approach [32]. The predictions of instance level logistic regression can be used to estimate the bag level probability [33]. Their main assumption is each individual instance has the same effect on the bag's label which is independent of each other. Therefore, their approach is taking the average of individual instances and maximization of likelihood at the bag level. The collective approach is extended in [34, 35] with feature selection and their learning algorithm selects the subset of the most important features at the instance level that contributes to the bag level classification.

Under standard MIL assumption, one approach is Diverse Density method [5,6,36] which focuses on finding a point t whose neighborhood has the highest positive bags' instances' intensity. This intensity is calculated using the proportion of the instances of positive bags to the negative bags. Another approach focuses on the distances between positive bags and negative bags. These approaches do not make assumptions about instances and concentrate only on distances between bags. One example of this approach is the representation of each bag with the minimum and maximum values of its instances [37, 38].

In [29], instances are not labeled as positive or negative. Instead, instances can belong to a concept or non-concept and they can be the member of a positive or negative bag. Their assumption is modeling the positive bags by a mixture of concept and non-concept distributions. Negative bags are determined with the non-concept distributions and positive bags are determined with both concept and non-concept. A positive bag may not have any concept in this case and they indicate that they may add a constraint that adds a minimum concept threshold to the positive bags. To determine whether an instance belongs to a concept, they use traditional classification approaches similar to [39].

[7] adopt the lazy learning strategy, Bayesian-KNN and Citation-KNN in MIL problems. In lazy learning classes of test examples can be found using classes of

training examples. However, this technique deals with instances. Therefore, it cannot be used directly in MIL problems. To solve this problem, Hausdorff distance and k-NN approaches in MIL problems are used.

Two alternatives for bag level classification are summarized in [18]. One alternative is bag-space approach [8]. In this approach, distances between bags and instances using some distance function are defined. After calculating these distances, they are used to calculate bag to bag distances. Bag to bag distances are used in classification algorithms such as k-nearest neighbor. This approach is also applied for recommending web index pages based on user behavior [40]. Another bag level classification method is defining directly a kernel over bags. This kernel calculates the difference or similarity between the bags. Each bag is represented by a graph and used graph kernel with support vector machines in [3, 41]. The other alternative is using a single vector that is generated from all instances in a bag with some calculation and using this vector in classification.

All research summarized before propose instance level or bag level approaches to MIL problems. Instance level approaches have dimensionality problem. Bag level approaches overcome the problem of dimensionality but have the disadvantage of losing information that can be gathered from instances.

These disadvantages lead to another solution under MIL domain, embedding approaches. Dissimilarity is one of the most popular embedding approaches [42, 43]. It stands for the representation of an object by describing it relative to a set of reference objects. This definition enables instance level information to be kept in a single vector of dissimilarities for each bag. Thus, the possibility of using low dimensional instance level information and complete bag level information together make dissimilarity a popular solution method in MIL domain.

Each bag is represented as a dissimilarity vector to other bags in [44]. Therefore, instances can be represented as features. Several bag to bag dissimilarity definition

method is introduced:

(i) **Point set** This method defines each bag as a set and the maximum mismatch between bags is calculated as the maximum of the minimum distances between vectors.

(ii) **Distribution of instances** This method approximates each bag by a normal distribution with parameters μ and σ and defines a distance formula using these parameters

(iii) **Attributed graph** In this method instances are the nodes and relationships between the instances are the edges.

Instance level and bag level methods are combined in [11] by showing the connection between the two methods. Bag to bag dissimilarity measures based on pairwise dissimilarities between instances is introduced in [37]. Bag to bag dissimilarities are calculated by different distance measures including mahalanobis and earth mover's is used [45]. The procedure starts with the estimation of the probability densities of instances for each bag and then after obtaining a single bag distribution these distribution vectors are used. They also adopt instance-based methods using overall minimum, mean minimum and haussdorf distances. After defining dissimilarities, they make a k-nearest neighbor classifier to obtain the results. A new bag to prototype dissimilarity is introduced in [46] which combines two standard dissimilarity approaches, first one is determined by the number of training bags and the second one is determined by the number of instances in the set. Bags are labeled with more than two levels in [47]. They use the random forest to determine the class of an instance which is the fraction of trees in the forest that votes for label y given input x and it is only used in segmentation. They also summarize the whole concept of data generation like determining features of each segment (i.e. turning their data into MIL data). A deterministic annealing algorithm for SVM formulations of MIL problems is presented in [48].

The application of neural network based approaches to MIL problems has been drawing attention from several different domains in recent years [4, 49–51]. Well known

problems have also been reformulated for this particular purpose, such as common computer vision tasks like image classification [52], weakly supervised object detection [53, 54]; sequence predictions [55], sentiment analysis [56] and sound event detection [57].

Embedding approaches propose performance superiority in terms of prediction power. However, one of the main challenges in MIL domain is that they also need to handle computational complexity [8]. Numerous researches concentrated on reducing the computational cost. For example, [58] proposes a bundle algorithm to SVM based methods [2, 20, 59].

However, most of the aforementioned methods require many hyperparameters to be tuned. Parameter tuning and computational cost lead us to other domains that do not require exhaustive searches. Random forests which are ensembles of decision trees [60], overcome both problems. Therefore, they are fast and simple like decision trees. In addition to that, random forests outperform decision trees in terms of prediction power [60]. As a result of these advantages, MIL literature proposes decision tree and tree-based ensemble approaches. [61] proposes a decision tree algorithm on propositionalised MIL data. [62] extends this approach with the random forest application. A tree-based encoding method is proposed in [63] in which each bag is represented with instance frequencies. [64] introduced a random forest approach as an extension to [65], a decision tree based approach in which a priority queue is introduced into the tree construction. Even though several tree-based ensemble approaches can be found in the literature, the idea of prototype based learning in a tree-based ensemble is a novel approach.

One other challenge in MIL solution approaches is providing results without sacrificing interpretability [4]. Only a few approaches summarized are capable of generating interpretable results [4, 30]. Furthermore, developing a robust approach that works in several MIL problems is also challenging. Certain methods summarized in this chapter outperform specific problem cases and suffer other problem cases. The main objective of this study is to provide basic and interpretable approaches to MIL problems

with performance superiority and fast algorithm execution time. As mentioned before, MIL concentrates on discrete labels. However, when real values are needed to be estimated the problem should be formulated in a different domain. The formulation of MIL problems in the regression domain (e.g. age estimation [66]) is called MIR. The initial studies on MIR are the extensions of the aforementioned MIL approaches into real valued responses including Citation-KNN and Diverse Density [67].

The initial studies on MIR follow Prime Instance Regression (PIR) [12]. The main assumption of PIR is that there is a prime instance in each bag that determines the bag's label. Therefore, the researches focus on the determination of prime instance for each bag [68, 69]. In PIR, a regression model is fit based on the primary instance. However, similar to instance level approaches and standard MIL assumption, PIR misses valuable information that comes from other instances. Due to the nature of MIL, bag labels may come from multiple instances or collections of several instances. As a result, assuming the label comes from a single instance may result in poor algorithm performances.

Therefore, extensions are proposed to PIR. A collective approach that assigns a salience value to each instance determines the weights of instances on the bag's label [14, 70]. However, this approach cannot be generalized to new bags that the labels and instance relevance are unknown. Prime instance clusters that assume the bag's label comes from a prime cluster of instances instead of a prime instance is another extension to PIR [13]. [71] proposed an instance pruning approach to estimate the aerosol optical (AOD) depth in the weather. The approach prunes the noisiest instances at every iteration and finds the important instances to make predictions. These instances are used to estimate the bag labels and the final prediction is made by certain operations like the average or mean of the instance level predictions. As an extension, a probabilistic framework that considers all instances may carry useful information about bag labels [72].

Another direction of MIR literature is embedding approaches. These approaches map the instances to a new embedding space to utilize the valuable instance level information. Kernel mean embedding [73] is a way of summarizing the instance level information into a new embedding space [15, 16].

2.3. Data Sets

MIL approaches are tested on 71 common MIL data sets on which most of the state-of-art methods are tested. Musk [19], Mutagenesis [74] and Protein [75] are molecular activity prediction data sets. The details of the Musk data sets are given before. Molecules are bags and atoms that belong to these molecules are instances of the Mutagenesis problem. The chemical properties of these atoms are the features of instances. The objective of the Protein problem is the identification of the membership of a protein to a family of TrX proteins.

Fox, Tiger, Elephant [76], Corel [2] and UCSB Breast Cancer [77] are the image annotation data sets. Fox, Tiger, and Elephant data sets consist of animal images. Images represent bags and the patches represent instances of the problem. The bag is labeled as positive if the respective animal is found in the image. There are 20 different Corel data sets. These data sets describe the scenery specified in the name of the data set. Each category has an original picture and a transformed picture. Each bag is an image and each instance is a $4x4$ pixel blocks. Features of each instance is the average colors that shows luminance and chrominance information. The other features comes from the square root of energy in the high-frequency bands of the wavelet transforms. UCSB Breast Cancer is a set of Tissue Microarray (TMA) images that are labeled as malignant or benign.

Newsgroups [3] and Web Recommendation [40] are text classification data sets. Newsgroups data set is a collection of approximately 20.000 newsgroup posts and is partitioned into 20 different groups. Each group represents a different topic varies from electronics to politics. In the Web data set, each bag represents a web index

page. Instances are also web pages that the original pages link to. The objective of the problem is to find whether a user favors at least one web page that an index page links to or not.

There are 13 different Bird [47, 78] data sets that belong to the group of audio recording classification. Each problem aims to identify whether a particular bird exists in a recording. Information related to all data sets can be found in Table 2.1.

Table 2.1. MIL data sets.

Name	**Inst.**	**Min**	**Max**	**Features**	**+Bags**	**-Bags**
Musk 1 $\oplus$	476	2	40	166	47	45
Musk 2 $\oplus$	6598	1	1044	166	39	63
Mutagenesis 1 $\oplus$	10486	28	88	7	125	63
Mutagenesis 2 $\oplus$	2132	26	86	7	13	29
Protein $\oplus$	26611	35	189	8	25	168
Elephant $\ominus$	1391	2	13	230	100	100
Fox $\ominus$	1302	1	13	230	100	100
Tiger $\ominus$	1220	2	13	230	100	100
Corel, African $\ominus$	7947	2	13	9	100	1900
Corel, Antique $\ominus$	7947	2	13	9	100	1900
Corel, Battleships $\ominus$	7947	2	13	9	100	1900
Corel, Beach $\ominus$	7947	2	13	9	100	1900
Corel, Buses $\ominus$	7947	2	13	9	100	1900
Corel, Cars $\ominus$	7947	2	13	9	100	1900
Corel, Desserts $\ominus$	7947	2	13	9	100	1900
Corel, Dinosaurs $\ominus$	7947	2	13	9	100	1900
Corel, Dogs $\ominus$	7947	2	13	9	100	1900
Corel, Elephants $\ominus$	7947	2	13	9	100	1900
Corel, Fashion $\ominus$	7947	2	13	9	100	1900
Corel, Flowers $\ominus$	7947	2	13	9	100	1900
Corel, Food $\ominus$	7947	2	13	9	100	1900

Table 2.1. MIL data sets. (cont.)

Corel, Historical ⊖	7947	2	13	9	100	1900
Corel, Horses ⊖	7947	2	13	9	100	1900
Corel, Lizards ⊖	7947	2	13	9	100	1900
Corel, Mountains ⊖	7947	2	13	9	100	1900
Corel, Skiing ⊖	7947	2	13	9	100	1900
Corel, Sunset ⊖	7947	2	13	9	100	1900
Corel, Waterfalls ⊖	7947	2	13	9	100	1900
UCSB Breast Cancer ⊖	2002	21	40	708	26	32
N.g. 1, alt.atheism ⊗	5443	22	76	200	50	50
N.g. 2, c.graphics ⊗	3094	12	58	200	50	50
N.g. 3, c.windows.misc ⊗	5175	25	82	200	50	50
N.g. 4, comp.pc.h. ⊗	4827	19	74	200	50	50
N.g. 5, comp.mac.h. ⊗	4473	17	71	200	50	50
N.g. 6, comp.windows.x ⊗	3110	12	54	200	50	50
N.g. 7, misc.forsale ⊗	5306	29	84	200	50	50
N.g. 8, rec.autos ⊗	3458	15	39	200	50	50
N.g. 9, rec.motorcycles ⊗	4730	22	73	200	50	50
N.g. 10, rec.sport.baseball ⊗	3358	15	58	200	50	50
N.g. 11, rec.sport.hockey ⊗	1982	8	38	200	50	50
N.g. 12, sci.crypt ⊗	4284	20	71	200	50	50
N.g. 13, sci.electronics ⊗	3192	12	58	200	50	50
N.g. 14, sci.med ⊗	3045	11	54	200	50	50
N.g. 15, sci.space ⊗	3655	16	59	200	50	50
N.g. 16, soc.rel.christian ⊗	4677	21	71	200	50	50
N.g. 17, politics.guns ⊗	3558	13	59	200	50	50
N.g. 18, politics.mideast ⊗	3376	15	55	200	50	50
N.g. 19, politics.misc ⊗	4788	21	75	200	50	50
N.g. 20, religion.misc ⊗	4606	25	79	200	50	50

Table 2.1. MIL data sets. (cont.)

Web recommendation 1 ⊗	2212	4	131	5863	17	58
Web recommendation 2 ⊗	2212	5	200	6519	18	57
Web recommendation 3 ⊗	2212	5	200	6306	14	61
Web recommendation 4 ⊗	2291	4	200	6059	55	20
Web recommendation 5 ⊗	2546	5	200	6407	61	14
Web recommendation 6 ⊗	2462	4	200	6417	59	16
Web recommendation 7 ⊗	2400	4	200	6450	39	36
Web recommendation 8 ⊗	2183	4	200	5999	35	40
Web recommendation 9 ⊗	2321	5	200	6279	37	38
Birds, Brown creeper ⊘	10232	2	43	38	197	351
Birds, Chestnut-b. ch. ⊘	10232	2	43	38	117	431
Birds, Dark-eyed junco ⊘	10232	2	43	38	20	528
Birds, Hammonds fly. ⊘	10232	2	43	38	103	445
Birds, Hermit thrush ⊘	10232	2	43	38	15	533
Birds, Hermit warbler ⊘	10232	2	43	38	63	485
Birds, Olive-sided fly. ⊘	10232	2	43	38	90	458
Birds, Pacific slope fly. ⊘	10232	2	43	38	165	383
Birds, Red-breasted nut. ⊘	10232	2	43	38	82	466
Birds, Swainsons thrush ⊘	10232	2	43	38	79	469
Birds, Varied thrush ⊘	10232	2	43	38	89	459
Birds, Western tanager ⊘	10232	2	43	38	46	502
Birds, Winter Wren ⊘	10232	2	43	38	109	439
⊕ molecular activity prediction, ⊖ image annotation						
⊗ text classification, ⊘ audio recording classification						

Multiple Instance Regression approaches are tested on Crop Yield [14] data sets. Crop Yield data set consists of wheat and corn yields in Kansas and California between 2001 and 2005. There are 525 bags in the data set and each bag represents a county in the data set. There are 100 instances in each bag. Each instance is 250m x 250m

of randomly selected pixels. The bag labels are the crop yields in bushels per acre that are reported by the U.S. Department of Agriculture (USDA). Instances have 92 features that are reflectance measures of red and infrared wavelengths. Each band has 46 observations that are measured every eight days at the same pixel point.

3. PROTOTYPE LEARNING IN MULTIPLE INSTANCE LEARNING

This chapter provides the details of Prototype Learning (PL). The work proposes a simple model that provides a prototype generator from a given MIL data set. We aim to find out prototypes in the feature space to map the collection of instances (i.e. bags) to a distance feature space and simultaneously learn a linear classifier for MIL. Another advantage of prototypical networks is that they are commonly used in the machine learning domain to facilitate interpretability. Our experiments on classical MIL benchmark data sets demonstrate that the proposed framework is an accurate and efficient classifier compared to the existing approaches.

3.1. Introduction

The model has two main objectives: The first one is learning the feature vectors or prototypes that are maximally predictive of the bag class after finding an embedding in the distance space. A simple illustration of the idea is presented in Figure 3.1 and Figure 3.2. Suppose there are two positive and two negative bags each of which has two instances in the two-dimensional feature space shown in Figure 3.1. We aim to identify prototypes such that the bags are linearly separable when each bag is represented by its minimum distance to each prototype. Figure 3.2 represents the bags in the new feature space. In other words, the proposed model is optimized over both the linear classifier parameters and the prototypes. An overview of architecture can be seen in Figure 3.3. Depending on the application, our proposal is flexible in generating the average and maximum type of features that are commonly used in multiple instance learning domain [44].

As illustrated in Figure 3.3, this method focuses on the dissimilarities between the bags and the prototypes. The main aim is to search for D prototypes which are generally smaller than the number of bags, M. Let P_d is a vector with length L. Feature

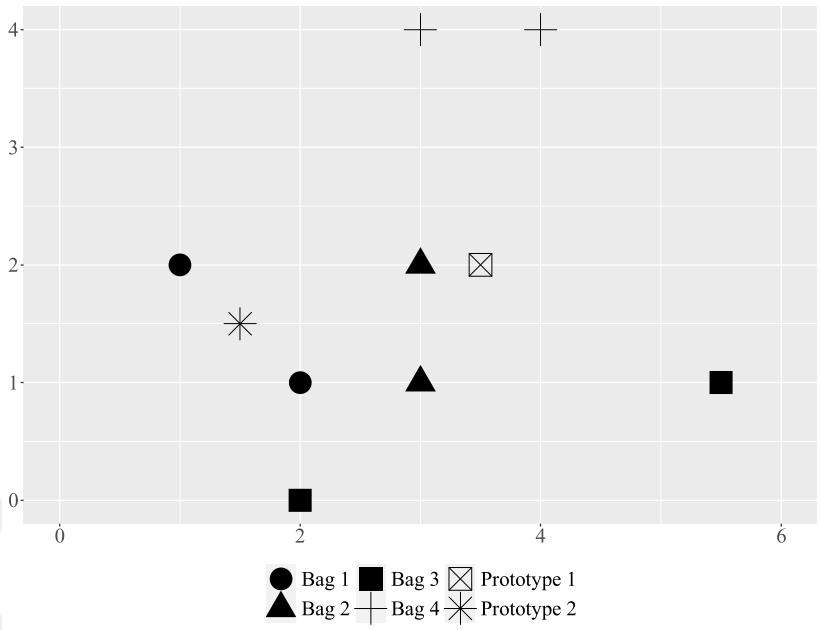


Figure 3.1. Bag 1 and Bag 2 are negative labeled bags. Bag 3 and Bag 4 have positive labels. All bags and prototypes have two features.

r of prototype d is defined as $P_{d,r}$. A bag B_i is represented based on its dissimilarity to each prototype. This way, a bag can be summarized with D features (i.e. distance of bag to each prototype). However, since a bag is composed of multiple instances and there is a need to define the distance between a prototype and instances. Euclidean distance of prototype d to instance X_{ij}, $Dist(X_{ij}, P_d)$, can be defined as

$$Dist(X_{ij}, P_d) = \sqrt{\sum_{r=1}^{L}(X_{ij,r} - P_{d,r})^2}. \tag{3.1}$$

3.2. Distance Feature Extraction

Similar to instance-level approaches in MIL problems, the proposed model also needs to pool information that is extracted from instances in a given bag with a potentially variable number of instances. To be more specific, for a given bag after the

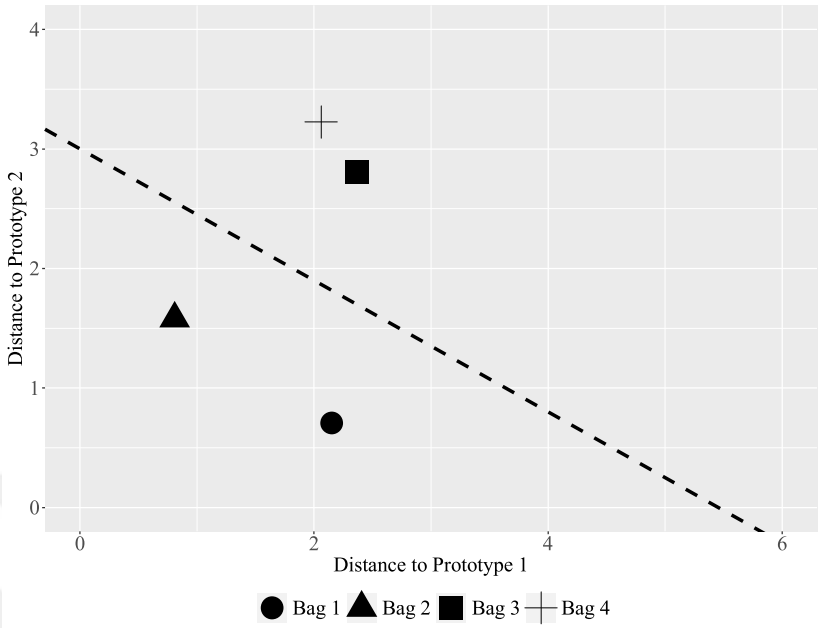


Figure 3.2. Distances of bags to prototypes. A good prototype candidate separates negative and positive labeled bags.

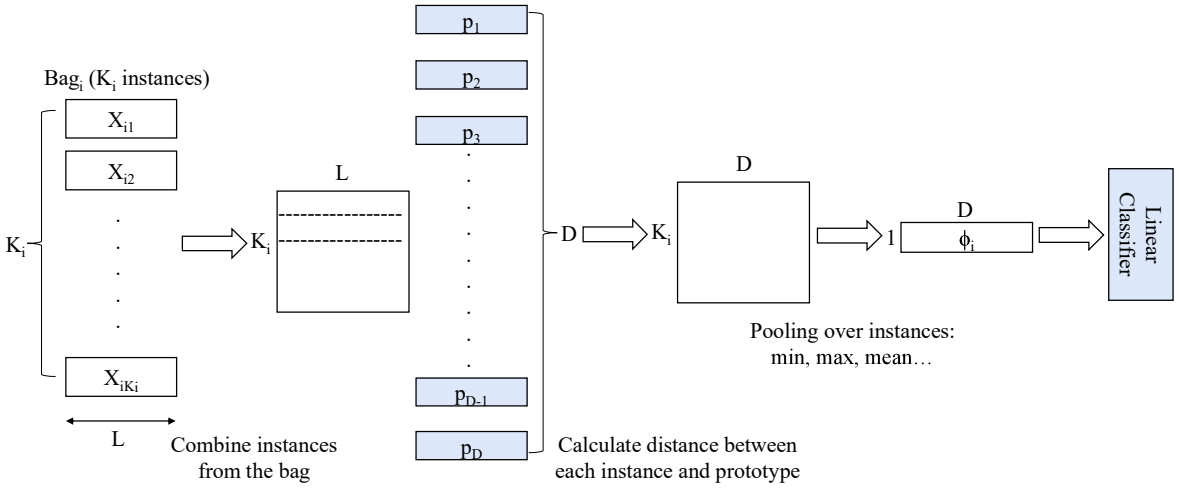


Figure 3.3. Overview of PL architecture. The blue color shows the variables, which the model will be optimized over. L: Number of features in an instance, constant; D: Number of prototypes, hyperparameter; K_i: Number of instances bag i, varies between different bags

distance from each instance to each prototype is calculated, the model needs to aggregate the information before being fed into a linear classifier. These pooling operations should be differentiable to be optimized with a gradient based approach [79]. The most basic and widely used pooling operators having these characteristics are min, mean and max operators [44]. These are also intuitively informative in our case, since we have distance metrics as features, such that these should provide information about defining characteristics of an instance, assuming the existence of prototypes which are described above.

Formally, the average distance between bag i and prototype d, $Dist(B_i, P_d)$, can be defined as

$$Dist(B_i, P_d) = \frac{1}{K_i} \sum_{i \in B_i} \sum_{j=1}^{K_i} Dist(X_{ij}, P_d). \tag{3.2}$$

Minimum and maximum distances can also be calculated very similarly to the below formulation.

The aim is to learn a logistic regression classifier to predict the binary positive or negative labels of bags, $\hat{y}$. In logistic regression, $\hat{y}$ is calculated as

$$\hat{y}_i = \beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id} \tag{3.3}$$

where β_0 is the intercept, β_d represent the weight in linear classifier that corresponds to d_{th} prototype. Φ_{id} is the d^{th} element of the output vector for bag i. Corresponds to the minimum distance of bag i to prototype d after layer normalization.

Logistic regression minimizes a cross-entropy loss function defined as

$$\mathcal{L}_{ce}(y, \hat{y}) = -yln(\sigma(\hat{y})) - (1 - y)ln(1 - \sigma(\hat{y})) \tag{3.4}$$

where $\mathcal{L}_{ce}(y, \hat{y})$ is the cross-entropy loss. In this definition, y is the label and $\hat{y}$ is the prediction. σ is the well-known sigmoid function that is defined as

$$\sigma(\hat{y}) = \frac{1}{1 + e^{-\hat{y}}}. \tag{3.5}$$

We can decompose the objective for each bag i and solve the nonlinear optimization problem using gradient descent. Decomposed cross-entropy loss will be denoted as $\mathcal{L}_{ce}(y_i, \hat{y}_i)$. A detailed formulation of the described method can be found in Section 3.3.

3.3. Model Formulation

A formulation for the discussed method is given in this section. The mathematical model is provided as

$$\min_{P,\beta} \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i, \hat{y}_i) + \lambda_w \left\|\beta\right\|_1 - \lambda_p \sum_{d=1}^{D}\sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{M}\sum_{d=1}^{D} \phi_{id} \tag{3.6}$$

$$\text{st} \qquad \Phi_{id}^{min} = \min_{X_{ij} \in B_i} Dist(X_{ij}, P_d) \tag{3.7}$$

$$\hat{y}_i = \sigma(\beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id}). \tag{3.8}$$

The objective function that is defined in Equation (3.6) is optimized over the prototypes, P, and the linear classifier weights, β. Different learning rates are used for prototypes and the classifier parameters, namely α_w for weights and α_p for prototypes. The formulation for *min* features is demonstrated in Equation (3.7). The *minimum* distance between bag i and prototype d is the minimum of pairwise distances between all instances of bag i to prototype d. Estimation of bag labels is expressed in Equation (3.8). The sigmoid is used for binary classification, but the same framework could easily be extended to multinomial case utilizing a softmax function.

The *minimum* distances in Equation (3.7) can be replaced with *maximum* or *average* distances which are defined as

$$\Phi_{id}^{max} = \max_{X_{ik} \in B_i} Dist(X_{ij}, P_d) \tag{3.9}$$

$$\Phi_{id}^{mean} = \frac{1}{K_i} \sum_{j=1}^{K_i} Dist(X_{ij}, P_d) \tag{3.10}$$

respectively. *maximum* distance in Equation (3.9) is the maximum of pairwise distances between all instances of bag i to prototype d. Similarly, Equation (3.10) defines the *average* distance as the average of pairwise distances between all instances of bag i to prototype d.

λ_w is the regularization parameter for linear classifier weights and λ_p is the regularization parameter for prototypes in the formulation. λ_d is the regularization parameter for the distances between the prototypes and the bags. Regularization of the distance between the prototypes prevents the algorithm to result in extreme values. Furthermore, better algorithm performance is obtained with this regularization. A more detailed analysis of regularization for prototypes will be shown in Section 3.6.

3.3.1. Feature Normalization

The distance features are prone to scale issues. This can cause problems with both gradient updates and the learning of linear classifier parameters. To overcome this, we adopt a similar approach to [80,81]. In other words, for each bag, we normalize the aggregated distance vector. Note that the information related to the "shape" of the distance features will be preserved under the normalization operation that is only recentering and rescaling, which is what we aim to achieve. Namely, each row in the transformed distance space is scaled to zero mean and unit variance which can be expressed as

$$\mu_i = \frac{1}{D}\sum_{d=1}^{D} \Phi_{id},\ \sigma_i = \sqrt{\frac{1}{D}(\Phi_i - \mu_i)^2} \tag{3.11}$$

$$\Phi_i = \frac{\Phi_i - \mu_i}{\sigma_i}. \tag{3.12}$$

Layer normalization is important because it stabilizes the issues that could occur during optimization due to the scale of distance features and it reduces the sensibility of the linear classifier to the scale of distances but it keeps the relative distance information. As a side benefit, we observe that as argued in [80], it speeds up the convergence.

In our setting, for a given training task we choose a fixed number of prototypes, D, of a fixed size, L, to be learned, we initialize these prototypes randomly. We combine each instance of length L in a given bag, which yields K_i (number of instances in bag i) vectors representing bag i. K_i is not constant between different bags, since each bag potentially has a different number of instances.

At each training step, we calculate the distance from each instance to the prototypes to extract distance features. Given these features, the model learns a classifier to predict the bag class. At each training step, we calculate the distance from each instance to the prototypes to extract distance features. Given these features, the model learns a classifier to predict the bag class. The approach summarized before applies to both multi class problems and binary classification problems. Details of the algorithm are given in Figure 3.5. There are two stopping criteria of the algorithm. The first criterion is the iterating the algorithm maximum number of iterations times. The second criterion is that the algorithm exits if AUC is not increased consecutive 3 epochs.

3.4. Interpretation

Interpretability of the solution is an important aspect of prototype learning. To demonstrate this interpretability, here we apply our approach to the MNIST MIL problem, which was introduced in [4]. MNIST database is an image database. The database has 60,000 training and 10,000 test images of handwritten digits [82]. A subset of images from MNIST database can be seen in Figure 3.4 [1].

In this case, each instance is an image, and each bag consists of images. The task is to find whether a target number exists in images in a bag. To keep things simple, we chose the number of prototypes to be 2. Examples of prototypes from two different runs can be seen in Figures 3.6 and 3.7. In this application, we only used *min* as the aggregator for better interpretation. For instance, looking at Figure 3.6, we see that the second prototype looks a lot like a 9, and the classifier found a negative coefficient for the minimum distance to this prototype. This indicates if the minimum

distance to this prototype is larger, the output probability will suffer. Moreover, since the first prototype has a positive coefficient, if the minimum distance of the bag to this prototype is larger, the output probability will be higher. The same analysis can be done for Figure 3.7.



Figure 3.4. MNIST image examples © 1998 IEEE [1].

3.5. Time Complexity

Let D be the number of prototypes, N be the number of data points in the dataset, K_i be the number of instances in the bag i, $K = \max_i K_i$, L be the number of features in an instance, E be the number of training epochs. Given an input i, a complete forward pass takes $\mathcal{O}(DK_iL + DK_i + D) = \mathcal{O}(DK_iL)$ which involves the computation of distances, the pooling operation and computing the output of the linear classifier. Investigating the training phase, we have that the back propagation and forward pass

has the same time complexities, and ultimately we obtain $\mathcal{O}(ENDKL)$, which is linear in all terms.

Definitions:

$\lambda_d, \lambda_p, \lambda_w$: Regularization parameters for distances between the prototypes and the bags, the prototypes and the linear classifier weights, α_w : Learning rate for classifier, α_p : Learning rate for prototypes, MaxIter: Number of iterations

Parameter tuning with inner cv;

foreach *Fold* **do**

  Initialize prototypes;

  BestAUC $\leftarrow 0, counter \leftarrow 0$;

  while $iter \leq MaxIter$ **do**

    foreach *i, d* **do**

      $\text{Dist}(\text{B}_i, P_d) \leftarrow 1/K_i \sum_{i \in B_i} \sum_{j=1}^{K_i} Dist(X_{ij}, P_d)$;

    end

    Calculate targets, AUC;

    if $AUC \geq BestAUC$ **then**

      $BestAUC \leftarrow AUC, counter \leftarrow 0$

    end

    $Loss \leftarrow \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i, \hat{y}_i) + \lambda_w \|\beta\|_1 - \lambda_p \sum_{d=1}^{D} \sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{M} \sum_{d=1}^{D} \phi_{id}$;

    Run Optimizer [83]: *Minimize Loss, Update prototypes with* α_p*, Update classifier weights with* α_w*;*

    if *Iter % Learning Rate Update = 0* **then**

      $\alpha_p \leftarrow \alpha_p/2, \alpha_w \leftarrow \alpha_w/2$

    end

  end

end

Figure 3.5. PL algorithm.

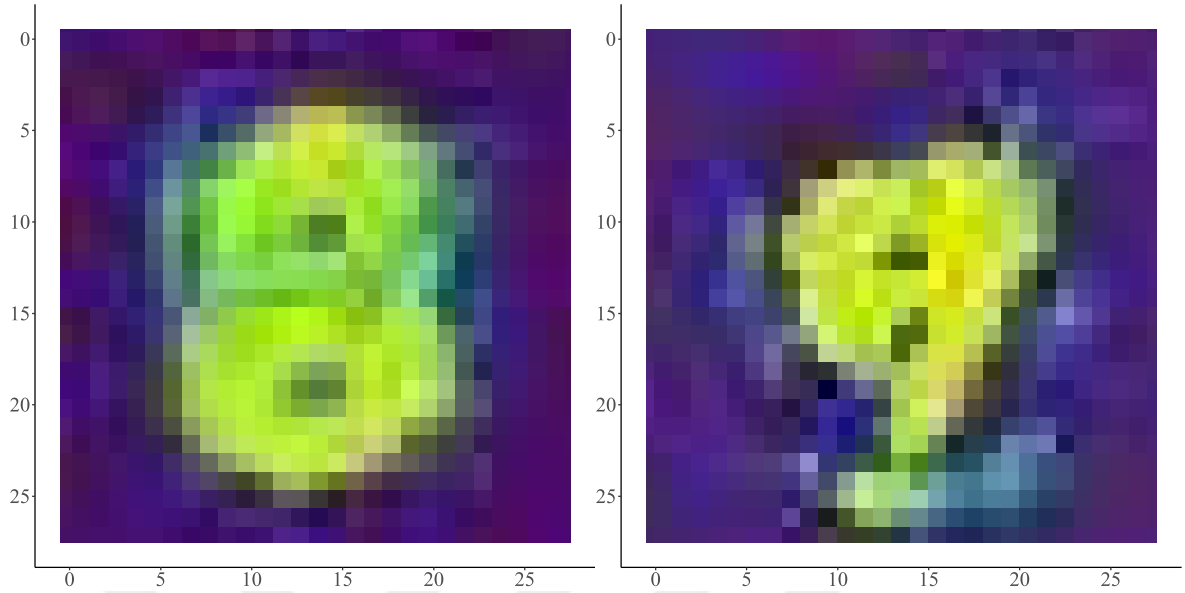


Figure 3.6. Prototypes of finding the 9 problem. Classifier coefficients of the prototypes: Left: 0.328, right: -0.359.

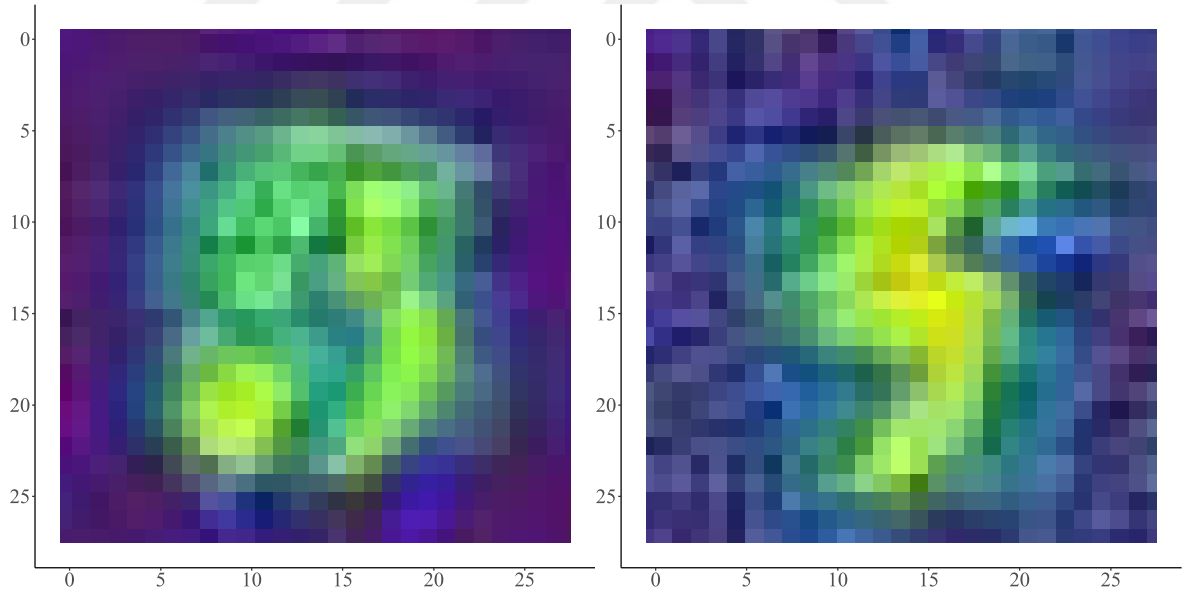


Figure 3.7. Prototypes of finding the 5 problem. Classifier coefficients of the prototypes: Left: 0.265, right: -0.359.

3.6. Experimental Results

MIL literature has common data sets that are generally used to report algorithm performances. There are 71 data sets that vary from molecular activity prediction to image annotation. The solution approach is tested on all of these data sets. The

details of these datasets can be found in Section 2.3. We repeat stratified 10-fold cross-validation five times. Randomly generated cross-validation indices and results of the benchmark models are taken from [84]. Considered benchmarks are APR (axis-parallel rectangles) [19], CCE (constructive clustering based ensemble) [85], Citation - KNN (citation k-nearest neighbor) [7], D_{maxmin}, $D_{meanmin}$, D_{minmin} (MIL with bag dissimilarities) [44], MILES (multiple instance learning via embedded instance selection) [2], miFV (MIL based on the Fisher Vector representation) [86], and QP-MIL (Quadratic Programming based MIL) [63]. For all experiments, initial prototypes are generated randomly, and logistic regression is used as the default classifier. The model was implemented in PyTorch [87]. Adam optimizer from [83] was used.

The parameters were tuned using Bayesian optimization [88]. Regularization parameters were searched between 0.00005 and 0.005. Boundaries of learning parameters were 0.00001 and 0.01. The number of prototypes was selected among $\{2, 4, 6, 8, 10\}$. We applied stepwise learning rate decay, namely in every 40 epochs, we decreased the learning rate to half. L2 regularization was applied to prototypes, and L1 regularization was applied to the classifier parameters. Layer normalization is applied as in [80] to the distance features. Experiments were run on Windows Server 2016 operating system with 10.0 GB installed memory (RAM) and a 2.30 GHz Intel Xeon(R) Processor.

3.7. Classification Accuracy

Solution approach in this study outperforms or at least does as good as all other well-known methods in terms of classification accuracy. Area Under The Curve (AUC) is our primary performance measure. Receiver Operating Characteristic (ROC) Curve is a graph that is plotted with True Positive Rate (TPR) against False Positive Rate (FPR). AUC measures the area under ROC curve. AUC is used to show how successful the model makes classification, especially if there is high imbalance between the numbers of positive and negative bags. Comparison of PL and other well-known approaches can be found in Table 3.1.

Table 3.1. Results of the algorithms.

Avgerage of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
Musk1	0.772	0.882	0.871	0.920	0.945	0.931	0.896	0.874	**0.968**	0.909
Musk2	0.806	0.781	0.850	0.956	**0.976**	0.956	0.823	0.786	0.945	0.892
Mutagenesis1	0.501	0.833	0.827	0.820	0.851	0.766	**0.910**	0.909	0.855	0.788
Mutagenesis2	0.458	0.727	0.688	0.590	0.647	0.345	**0.882**	0.867	0.785	0.873
Protein	0.509	0.643	0.552	0.561	0.523	**0.876**	0.872	0.873	0.851	0.867
Elephant	0.728	0.854	0.882	0.876	0.936	0.915	0.886	0.882	**0.941**	0.936
Fox	0.585	0.623	0.569	0.457	0.612	0.704	0.630	0.655	0.677	**0.737**
Tiger	0.583	0.832	0.752	0.761	0.853	0.850	0.826	0.860	**0.901**	0.892
CorelAfrican	0.580	0.840	0.862	0.939	**0.967**	0.966	0.836	0.851	0.953	0.902
CorelAntique	0.595	0.722	0.747	0.877	**0.922**	0.910	0.769	0.801	0.871	0.894
CorelBattleships	0.569	0.912	0.866	0.972	**0.981**	0.966	0.876	0.889	0.958	0.972
CorelBeach	0.596	0.969	0.926	0.972	0.983	0.990	0.977	0.980	**0.993**	0.979
CorelBuses	0.591	0.963	0.848	0.962	0.973	**0.981**	0.944	0.944	0.960	0.960
CorelCars	0.608	0.878	0.835	0.930	**0.948**	0.916	0.856	0.864	0.912	0.905
CorelDesserts	0.571	0.922	0.866	0.950	**0.974**	0.966	0.915	0.943	0.969	0.956
CorelDinosaurs	0.551	0.897	0.802	0.948	**0.983**	0.982	0.882	0.916	0.964	0.959
CorelDogs	0.570	0.805	0.752	0.894	**0.919**	0.911	0.838	0.856	0.863	0.844

Table 3.1. Results of the algorithms. (cont.)

Avgerage of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
CorelElephants	0.643	0.893	0.870	0.966	**0.983**	0.970	0.858	0.884	0.968	0.923
CorelFashion	0.868	0.950	0.909	0.983	0.990	**0.991**	0.904	0.928	0.986	0.953
CorelFlowers	0.636	0.844	0.833	0.932	0.947	**0.953**	0.856	0.890	0.938	0.885
CorelFood	0.860	0.985	0.934	0.993	**0.998**	0.996	0.979	0.984	0.992	0.983
CorelHistorical	0.773	0.978	0.942	0.984	**0.998**	0.994	0.985	0.987	0.994	0.929
CorelHorses	0.613	0.809	0.746	0.861	**0.920**	0.917	0.798	0.815	0.860	0.859
CorelLizards	0.553	0.938	0.897	0.960	**0.980**	0.971	0.940	0.934	0.973	0.953
CorelMountains	0.981	0.988	0.979	0.998	**1.000**	0.999	0.988	0.993	0.998	0.963
CorelSkiing	0.509	0.875	0.770	0.956	**0.960**	0.953	0.867	0.873	**0.960**	0.923
CorelSunset	0.495	0.728	0.706	0.803	**0.837**	0.751	0.704	0.751	0.739	0.766
CorelWaterfalls	0.595	0.836	0.865	0.962	**0.975**	0.966	0.839	0.896	0.955	0.891
UCSBBreastCancer	0.569	0.644	0.706	0.725	0.831	0.791	0.823	0.848	0.888	**0.908**
Newsgroups1	0.500	0.792	0.803	0.905	**0.941**	0.500	0.667	0.654	0.933	0.837
Newsgroups2	0.508	0.660	0.638	0.895	**0.898**	0.554	0.718	0.614	0.831	0.870
Newsgroups3	0.500	0.622	0.584	**0.818**	0.810	0.500	0.650	0.660	0.778	0.763
Newsgroups4	0.508	0.678	0.636	0.822	**0.857**	0.479	0.704	0.663	0.820	0.845
Newsgroups5	0.488	0.609	0.585	**0.853**	0.852	0.559	0.649	0.638	0.812	0.822

Table 3.1. Results of the algorithms. (cont.)

Avgerage of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
Newsgroups6	0.500	0.742	0.732	0.865	**0.890**	0.572	0.676	0.650	0.849	0.882
Newsgroups7	0.496	0.648	0.633	0.752	0.790	0.547	0.645	0.654	**0.816**	0.742
Newsgroups8	0.498	0.692	0.570	0.840	**0.870**	0.460	0.713	0.623	0.818	0.764
Newsgroups9	0.500	0.806	0.818	0.348	0.326	0.500	0.699	0.616	0.850	**0.881**
Newsgroups10	0.500	0.747	0.821	**0.918**	0.914	0.476	0.624	0.652	0.880	0.747
Newsgroups11	0.492	0.794	0.714	**0.968**	0.958	0.460	0.667	0.634	0.923	0.810
Newsgroups12	0.510	0.756	0.806	**0.868**	0.840	0.466	0.634	0.642	0.865	0.786
Newsgroups13	0.500	0.528	0.621	0.932	0.946	0.500	0.468	0.632	**0.949**	0.921
Newsgroups14	0.500	0.777	0.780	0.922	**0.942**	0.465	0.666	0.641	0.888	0.728
Newsgroups15	0.500	0.790	0.763	0.874	0.905	0.517	0.621	0.644	**0.921**	0.858
Newsgroups16	0.500	0.787	0.773	0.879	**0.898**	0.509	0.640	0.627	0.822	0.882
Newsgroups17	0.494	0.766	0.692	0.818	**0.874**	0.531	0.680	0.620	0.796	0.760
Newsgroups18	0.500	0.825	0.845	0.833	**0.874**	0.463	0.688	0.666	0.852	0.793
Newsgroups19	0.502	0.831	0.766	0.785	0.802	0.561	0.642	0.638	0.814	**0.856**
Newsgroups20	0.498	0.770	0.646	**0.839**	0.834	0.443	0.625	0.642	0.813	0.726
Web1	0.547	0.788	0.614	0.411	0.634	0.788	0.826	**0.834**	0.655	0.829
Web2	0.522	0.473	0.445	0.501	0.474	0.521	**0.700**	0.693	0.537	0.691

Table 3.1. Results of the algorithms. (cont.)

Avgerage of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
Web3	0.600	0.603	0.649	0.501	0.708	0.608	**0.772**	0.767	0.661	0.733
Web4	0.575	0.834	0.707	0.548	0.799	0.629	0.817	**0.847**	0.627	0.794
Web5	0.540	0.612	0.506	0.507	0.711	**0.794**	0.731	0.776	0.552	0.678
Web6	0.581	0.621	0.520	0.494	0.525	0.543	0.722	0.727	0.650	**0.741**
Web7	0.586	0.592	0.675	0.600	0.690	0.670	0.690	0.688	0.545	**0.744**
Web8	0.552	0.575	0.498	0.579	0.409	0.366	0.652	0.650	0.530	**0.710**
Web9	0.595	0.631	0.599	0.497	0.735	0.687	0.730	0.741	0.503	**0.767**
BrownCreeper	0.592	0.945	0.883	0.729	0.899	0.927	0.989	**0.992**	0.991	0.926
Chestnut-backedChickadee	0.520	0.827	0.802	0.831	0.853	0.749	0.898	0.910	**0.917**	0.844
Dark-eyedJunco	0.683	0.667	0.594	0.695	0.856	0.870	0.938	**0.947**	0.932	0.786
HammondsFlycatcher	0.534	0.992	0.884	0.718	0.944	0.883	0.999	0.999	**1.000**	0.996
HermitThrush	0.668	0.488	0.535	0.570	0.578	0.892	0.808	0.836	**0.903**	0.750
HermitWarbler	0.593	0.818	0.697	0.735	0.781	0.926	0.981	0.979	**0.984**	0.880
Olive-sidedFlycatcher	0.644	0.878	0.794	0.853	0.896	0.924	**0.969**	0.963	0.967	0.933
PacificslopeFlycatcher	0.531	0.836	0.699	0.723	0.754	0.769	0.957	**0.958**	0.943	0.831
Red-breastedNuthatch	0.583	0.872	0.771	0.803	0.876	0.877	0.979	**0.982**	0.971	0.888
SwainsonsThrush	0.519	0.826	0.687	0.782	0.767	0.884	0.968	**0.981**	0.977	0.771

Table 3.1. Results of the algorithms. (cont.)

Avgerage of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
VariedThrush	0.584	0.869	0.782	0.751	0.840	0.940	**1.000**	**1.000**	0.997	0.949
WesternTanager	0.696	0.878	0.755	0.475	0.849	0.824	**0.988**	0.982	0.973	0.897
WinterWren	0.598	0.965	0.907	0.944	0.932	0.854	**0.993**	0.990	0.988	0.928
Performance Rank	10	9	8	5	2	7	6	4	1	3

We apply the procedure in [89] to compare the results of different approaches. Friedman Test is a non-parametric test and concentrates on the average ranks. In our test, Null Hypothesis indicates that average ranks of all different approaches are same. However, when we calculate the test statistic, p-value is less than α at significance level 5%. We reject Null Hypothesis and conclude that average ranks of all different approaches are not same. Therefore, we continue with Nemenyi Test [90]. In Nemenyi Test, we calculate a critical difference (CD). If average rank of two approaches is greater than Nemenyi CD value, we can conclude that performances of two approach is significantly different than each other. Results of Nemenyi Test can be found in Figure 3.8. The best three performing approaches are QP-MIL, $\mathrm{D}_{meanmin}$ and PL in terms of average rank. However, there is no significant difference between these three methods. The results of the algorithms are aggregated to the problem categories and average AUC values are reported in Table 3.2. When the average AUC values are analyzed, QP-MIL and PL outperform all other methods with an average of 0.865 and 0.853 respectively.

To analyze the results better, PL's pairwise comparison to other methods is visualized. PL outperforms APR all data sets except CorelMountains as Figure 3.9 shows. Even average AUC value of APR over all Corel data sets is 0.638, the approach outperforms in Mountains data set. Mountains data set has the highest average AUC value among all approaches. However, even with the data characteristics, PL is expected to perform better than APR.

PL's comparison to CCE and Citation KNN can be found in Figure 3.10 and Figure 3.11 respectively. PL outperforms both methods considering average rank and average AUC. The performance of PL is very close to other methods' performances on the data sets in which PL is outperformed.

PL outperforms $D_{maxmin}, D_{meanmin}$, and D_{minmin} on Bird, Web, Animals, and Mutagenesis data sets. The average AUC values of PL are significantly higher on these data sets. For example, the average AUC difference between PL and the three methods

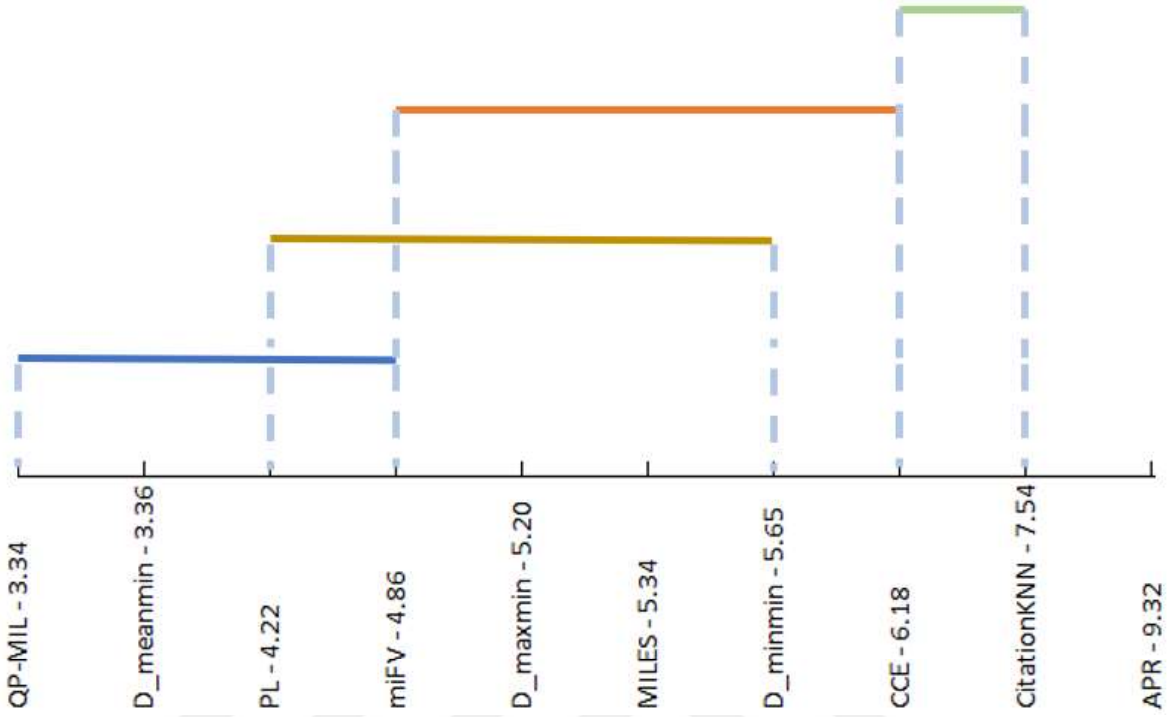


Figure 3.8. Mean AUC Performances. Numbers indicate the average ranks of the approach.

is 0.153 on Web data sets. These three data sets outperform PL on Newgroups and Corel data sets. However, the difference is smaller when compared to the data sets that PL outperforms. The pairwise comparisons D_{maxmin}, $D_{meanmin}$, and D_{minmin} to PL are shown in Figure 3.12, Figure 3.13, and Figure 3.14 respectively. PL outperforms MILES on all data sets except Birds and Mutagenesis data sets. Similarly, PL's performance is better than miFV's performance on all data sets except Birds, Mutagenesis, and Protein. The comparison of these two data sets to PL can be found in Figure 3.15 and Figure 3.16 respectively. PL's pairwise comparison with QP-MIL can be found in Figure 3.17. PL outperforms QP-MIL on Web data sets. However, the performance of QP-MIL is better than PL on Birds data sets.

Table 3.2. Results of the algorithms based on problem categories.

Average of AUC	**APR**	**CCE**	**Cit. KNN**	**Dmax**	**Dmean**	**Dmin**	**MILES**	**miFV**	**QP-MIL**	**PL**
Musk	0.789	0.831	0.861	0.938	**0.960**	0.944	0.859	0.830	0.957	0.901
Mutagenesis	0.480	0.780	0.757	0.705	0.749	0.555	**0.896**	0.888	0.820	0.831
Protein	0.509	0.643	0.552	0.561	0.523	**0.876**	0.872	0.873	0.851	0.867
Animals	0.632	0.769	0.734	0.698	0.800	0.823	0.781	0.799	0.840	**0.855**
Corel	0.638	0.887	0.848	0.942	**0.962**	0.952	0.881	0.899	0.940	0.920
UCSB	0.569	0.644	0.706	0.725	0.831	0.791	0.823	0.848	0.888	**0.908**
Newsgroups	0.500	0.731	0.711	0.837	0.851	0.503	0.654	0.640	**0.851**	0.814
Web	0.566	0.637	0.579	0.515	0.632	0.623	0.738	**0.747**	0.585	0.743
Birds	0.596	0.835	0.753	0.739	0.833	0.871	0.959	0.963	**0.965**	0.875
Avg	0.579	0.785	0.745	0.796	0.844	0.749	0.808	0.811	**0.865**	0.853

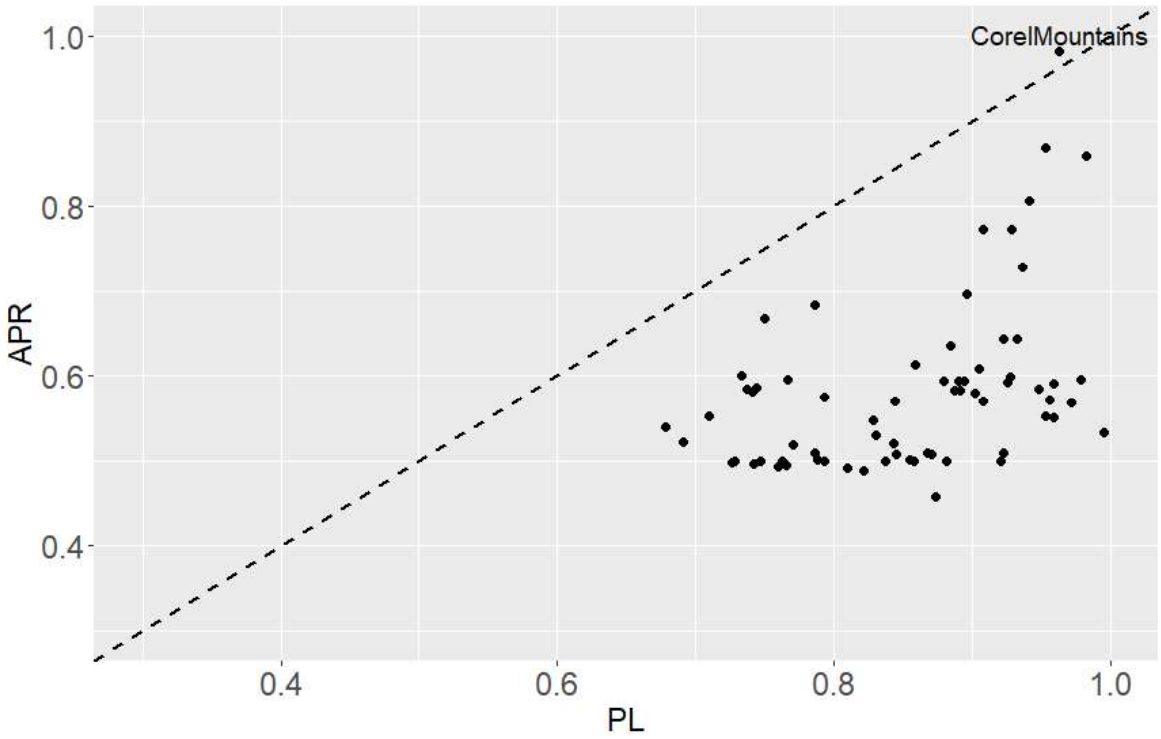


Figure 3.9. AUC comparison of PL and APR on benchmark data sets.

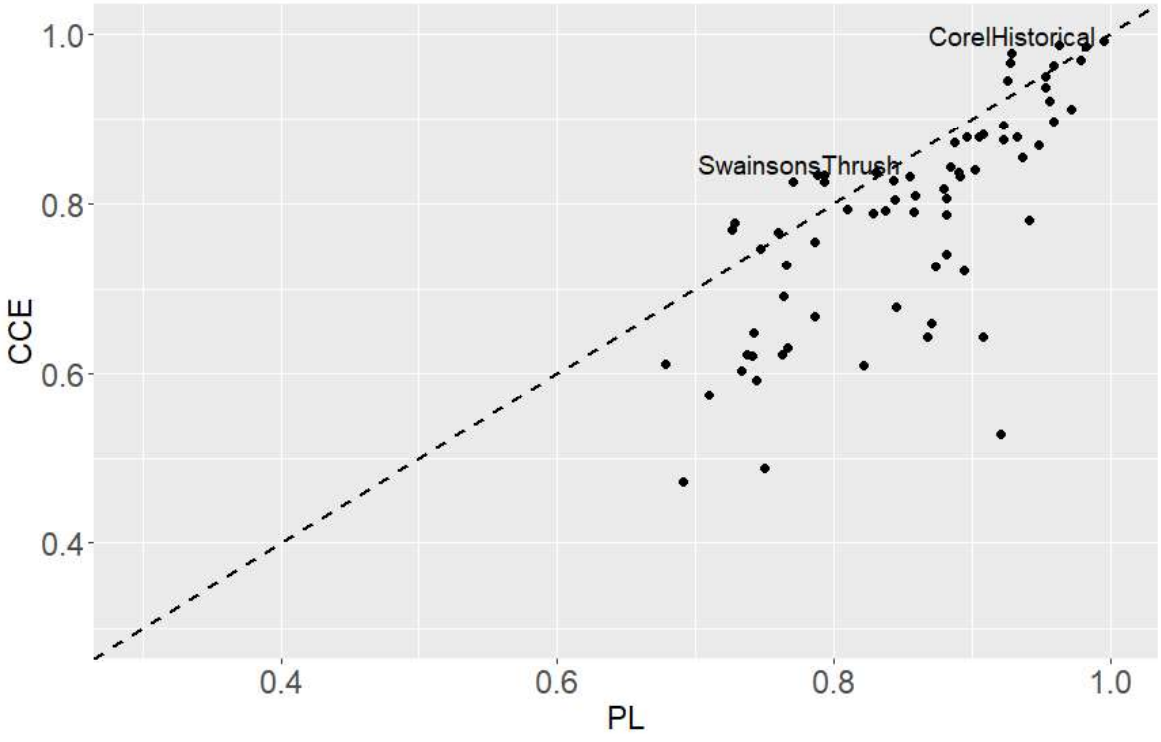


Figure 3.10. AUC comparison of PL and CCE on benchmark data sets.

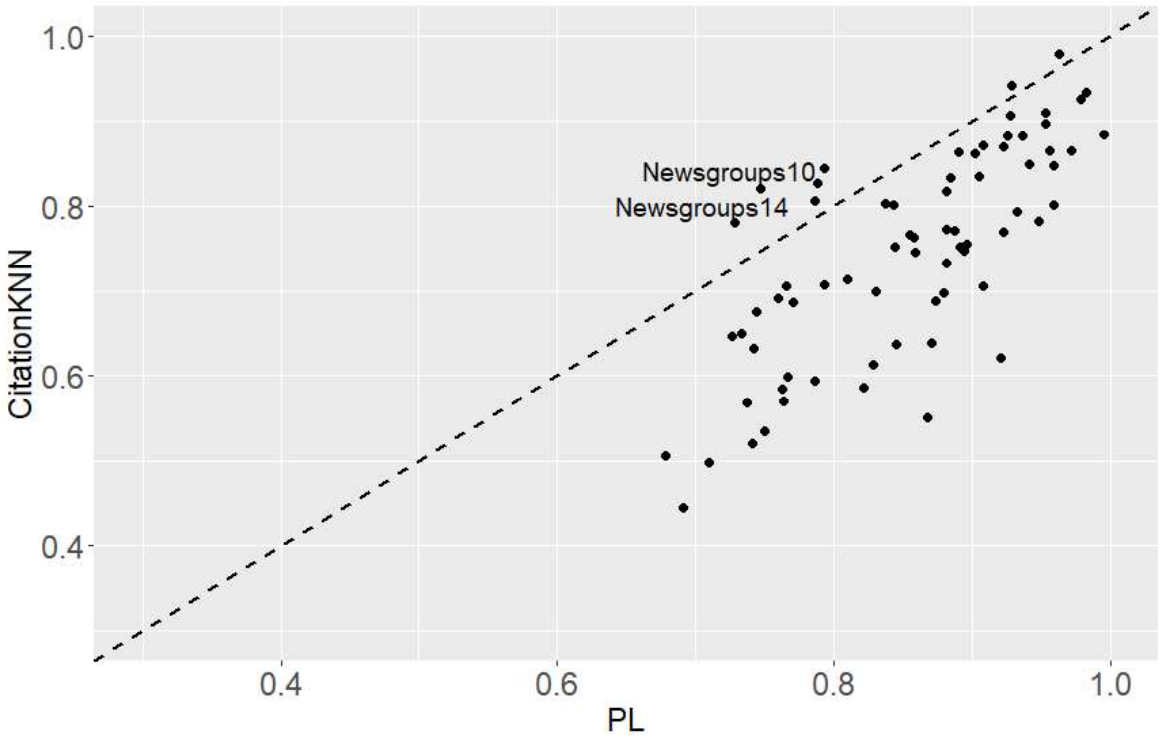


Figure 3.11. AUC comparison of PL and Citation-KNN.

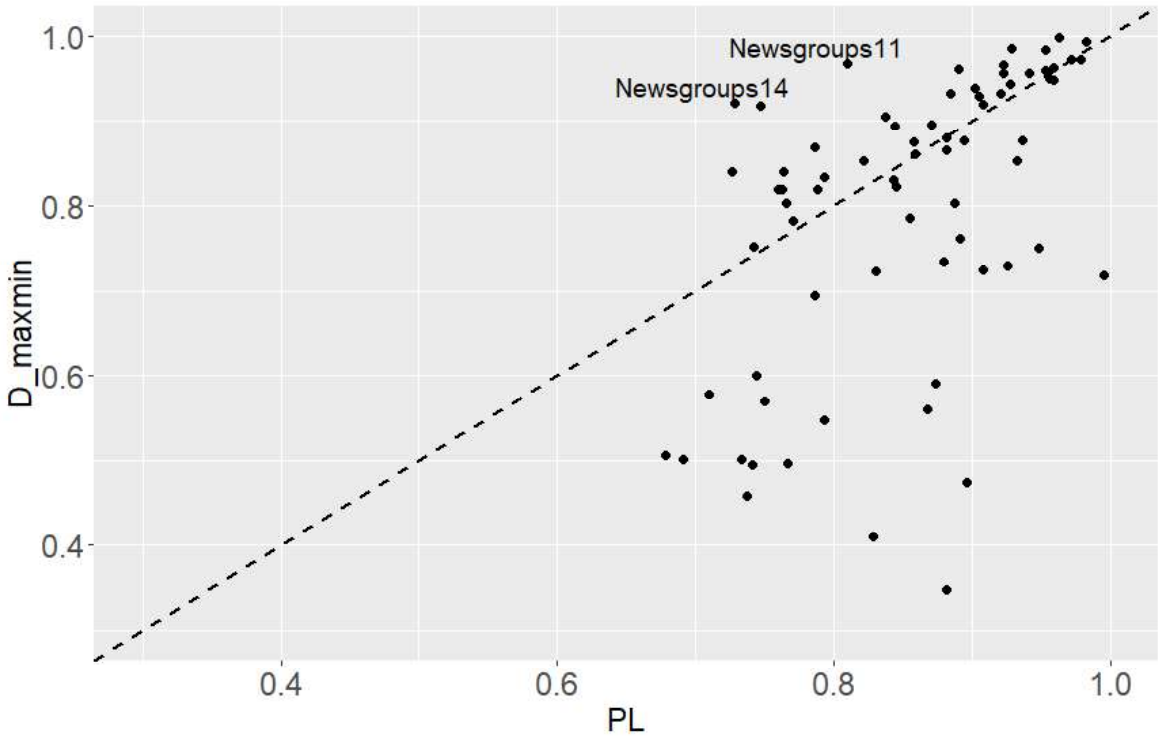


Figure 3.12. AUC comparison of PL and D_{maxmin} on benchmark data sets.

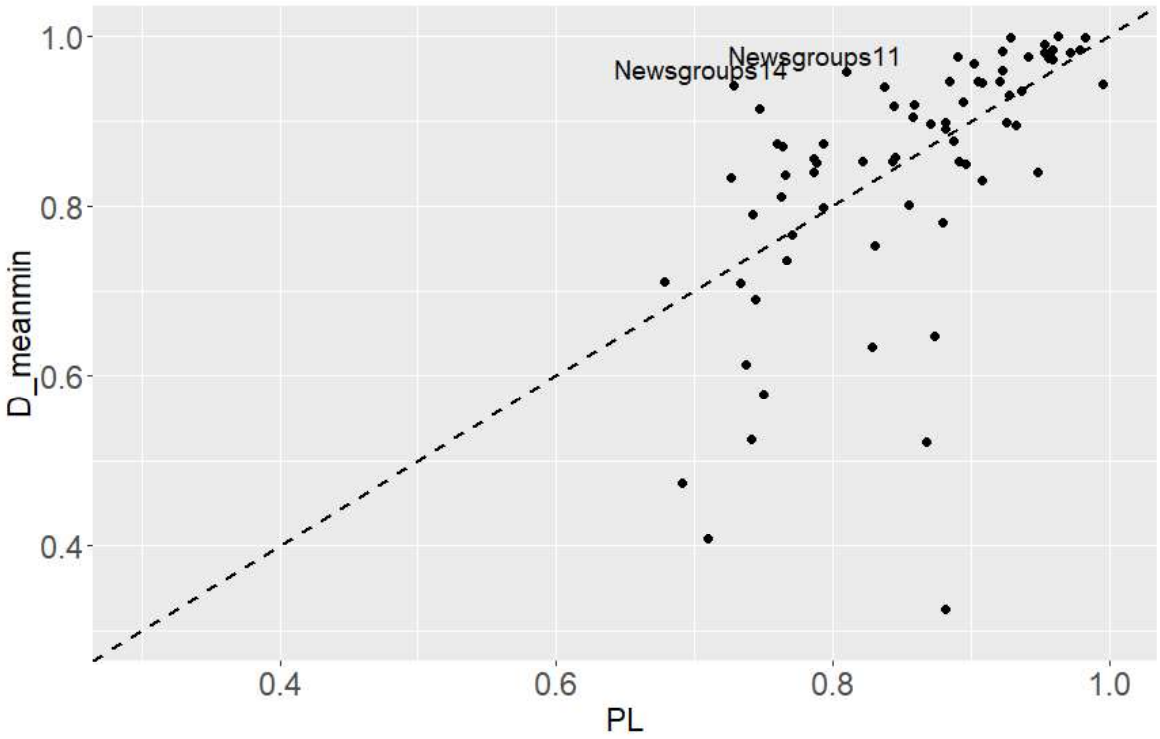


Figure 3.13. AUC comparison of PL and $D_{meanmin}$ on benchmark data sets.

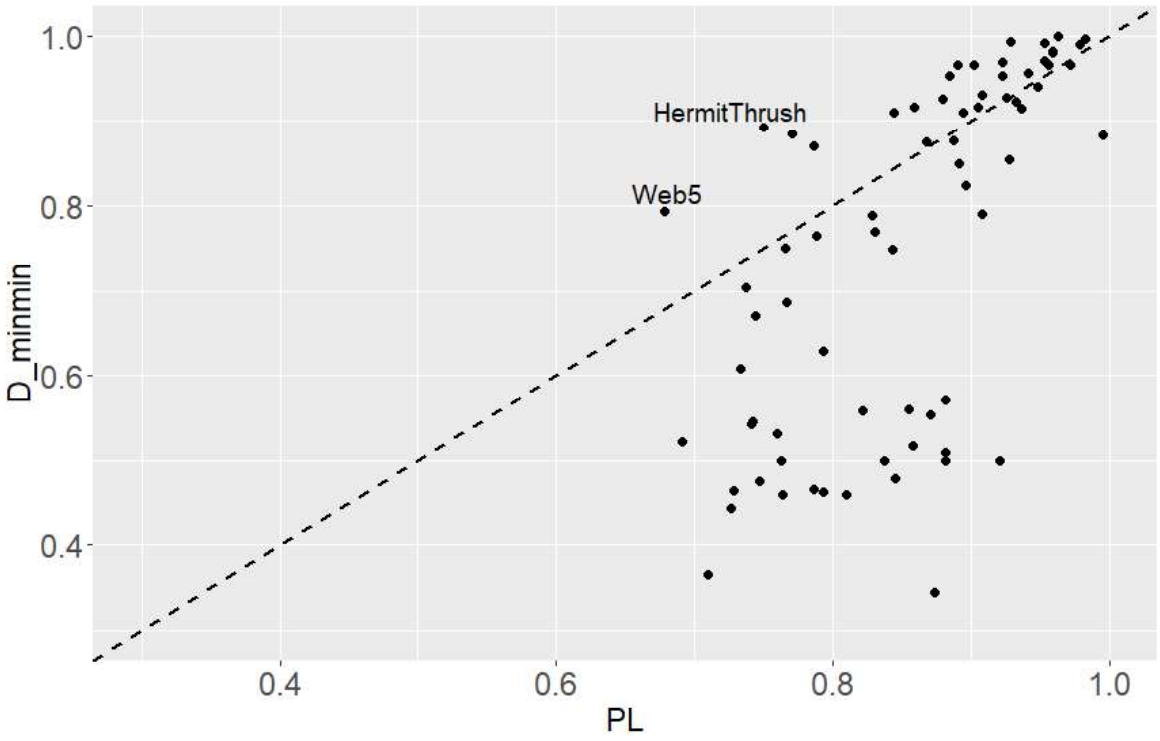


Figure 3.14. AUC comparison of PL and D_{minmin} on benchmark data sets.

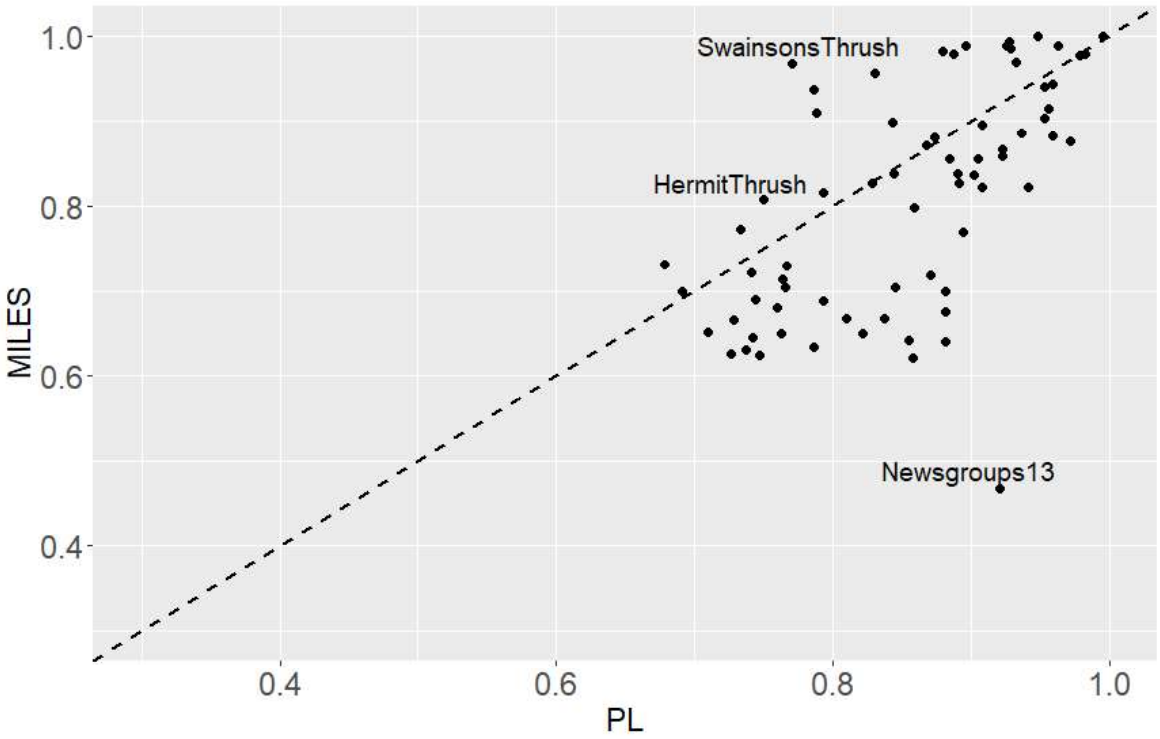


Figure 3.15. AUC comparison of PL and MILES on benchmark data sets.

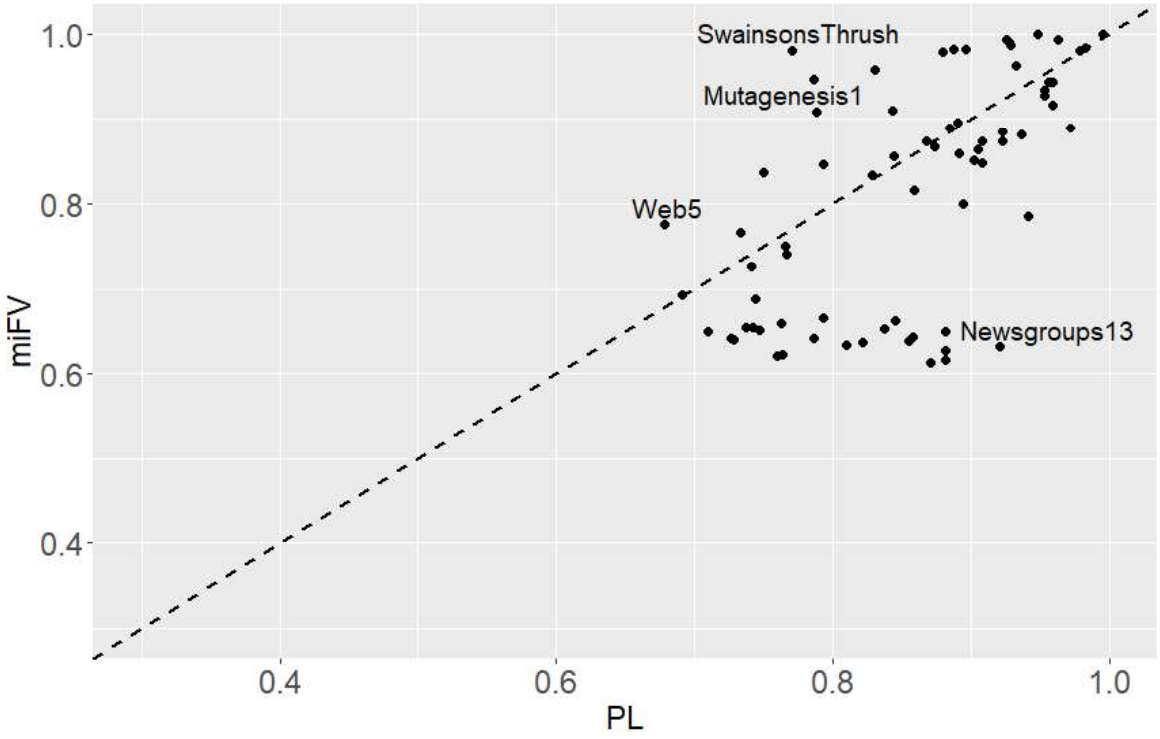


Figure 3.16. AUC comparison of PL and miFV on benchmark data sets.

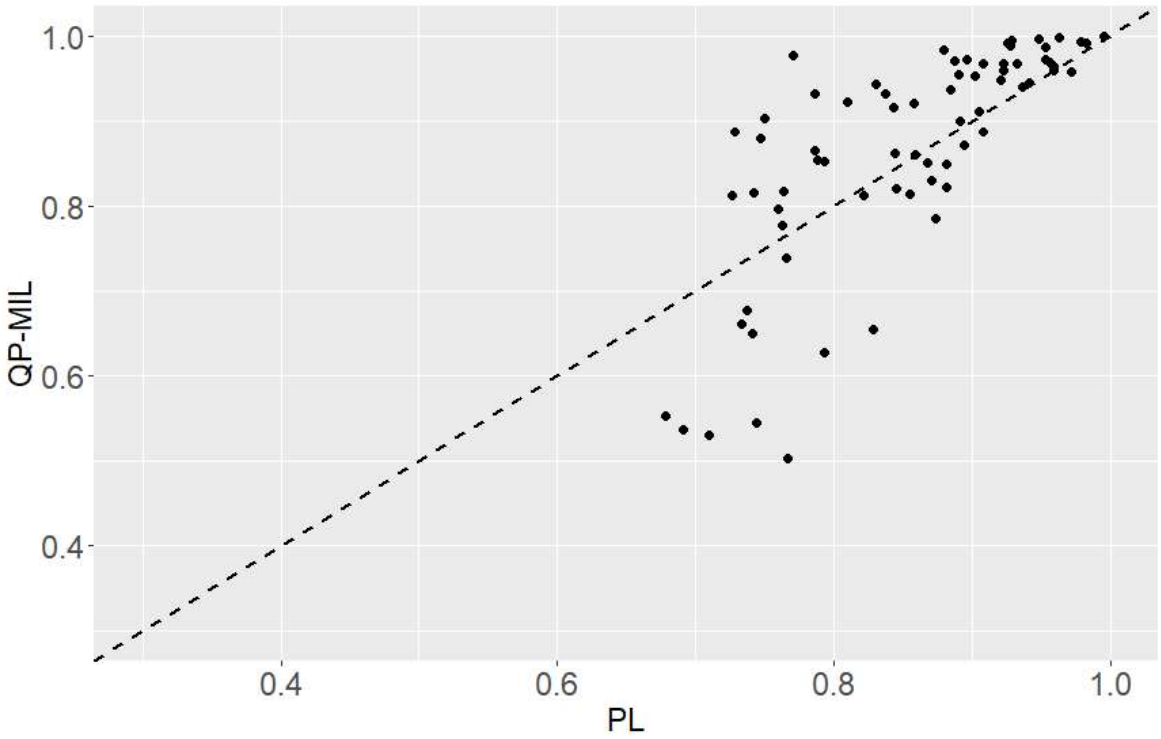


Figure 3.17. AUC comparison of PL and QP-MIL on benchmark data sets.

To analyze the run time of the algorithm, the average run time of each replicate is reported in Table 3.3. As a result of all experiments, we can conclude that average run times are highly correlated with the number of features and number of bags as expected. When run times are analyzed for several data sets, it can be concluded that Corel data sets are completed 3408 seconds on average per replicate. Having more bags than any other data set in experiments, Corel data sets have the longest run time. The average run time per replicate of Newsgroups data sets is 802.9 seconds. However, hyperparameter tuning of four Newsgroups data sets takes longer than others. If Newsgroups 6-8 and Newsgroups 11 are excluded, the average run time decreases to 670.1 seconds. Web data sets has the highest number of features. Even the number of bags is low in Web data sets, the number of features increases the run time. The smallest data sets (e.g. Musk) in terms of number of features, number of bags and number of instances has the lowest run time.

Table 3.3. Average run time of the algorithm (in seconds).

Data Set	**Run Time**	**Data Set**	**Run Time**
BrownCreeper	1248	Newsgroups5	728
Chestnut-backedChickadee	1057	Newsgroups6	1139
CorelAfrican	3500	Newsgroups7	1272
CorelAntique	3026	Newsgroups8	1766
CorelBattleships	3220	Newsgroups9	660
CorelBeach	3199	Newsgroups10	618
CorelBuses	3178	Newsgroups11	1150
CorelCars	3314	Newsgroups12	643
CorelDesserts	3191	Newsgroups13	601
CorelDinosaurs	3239	Newsgroups14	600
CorelDogs	4191	Newsgroups15	599
CorelElephants	3429	Newsgroups16	617
CorelFashion	3643	Newsgroups17	653
CorelFlowers	3581	Newsgroups18	737
CorelFood	3468	Newsgroups19	724
CorelHistorical	3273	Newsgroups20	656
CorelHorses	3238	Olive-sidedFlycatcher	1214
CorelLizards	3296	PacificslopeFlycatcher	1177
CorelMountains	3182	Protein	1442
CorelSkiing	3436	Red-breastedNuthatch	1300
CorelSunset	4086	SwainsonsThrush	1106
CorelWaterfalls	3486	Tiger	778
Dark-eyedJunco	1097	UCSBBreastCancer	1279
Elephant	1210	VariedThrush	1109
Fox	1518	Web1	1219
HammondsFlycatcher	1086	Web2	1890
HermitThrush	1206	Web3	2384
HermitWarbler	1288	Web4	2789

Table 3.3. Average run time of the algorithm (in seconds). (cont.)

Musk1	715	Web5	3264
Musk2	713	Web6	3027
Mutagenesis1	884	Web7	3363
Newsgroups1	788	Web8	3336
Newsgroups2	760	Web9	3754
Newsgroups3	722	WesternTanager	1089
Newsgroups4	625	WinterWren	1095

3.8. Parameter Sensitivity

The algorithm's performance changes under different parameter configurations. Average AUC scores of different parameter settings are reported for the Musk2 dataset to show the size of the change, and the robustness of the algorithm. Ten-fold cross validation is repeated five times for each parameter setting.

Parameter sensitivity of learning rate of prototypes and weights are analyzed under the set of {0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.05} and {0.0001, 0.0005, 0.001, 0.005, 0.05} respectively. Related performances are reported in Figures 3.18 and 3.19. If the learning rate is increased to 0.001 and above, the average AUC suffers. Decreasing learning rates below 0.0001 does not increase performance even if it brings high computational cost and slow convergence.

Experiments with number of prototypes $\in \{2, 4, 6, 8, 10\}$ can be found in Figure 3.20. The best performance is obtained when the number of prototypes is 10. However, less number of prototypes shows poor performance. The performance of PL is robust to the increasing number of prototypes due to the regularization applied to the weights.

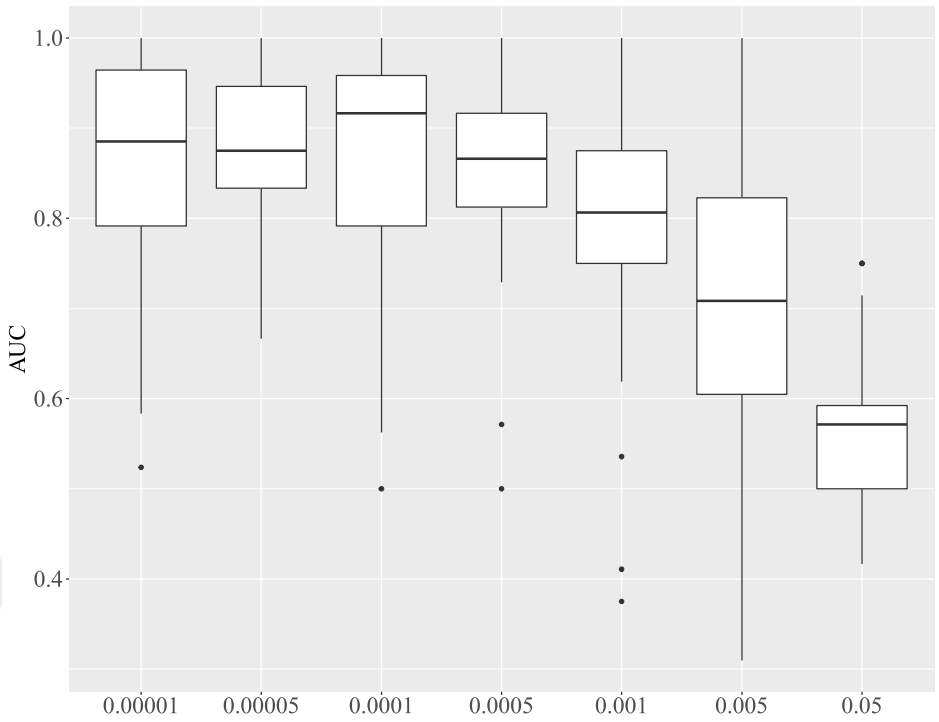


Figure 3.18. Average AUC performance under different learning rates of prototypes.

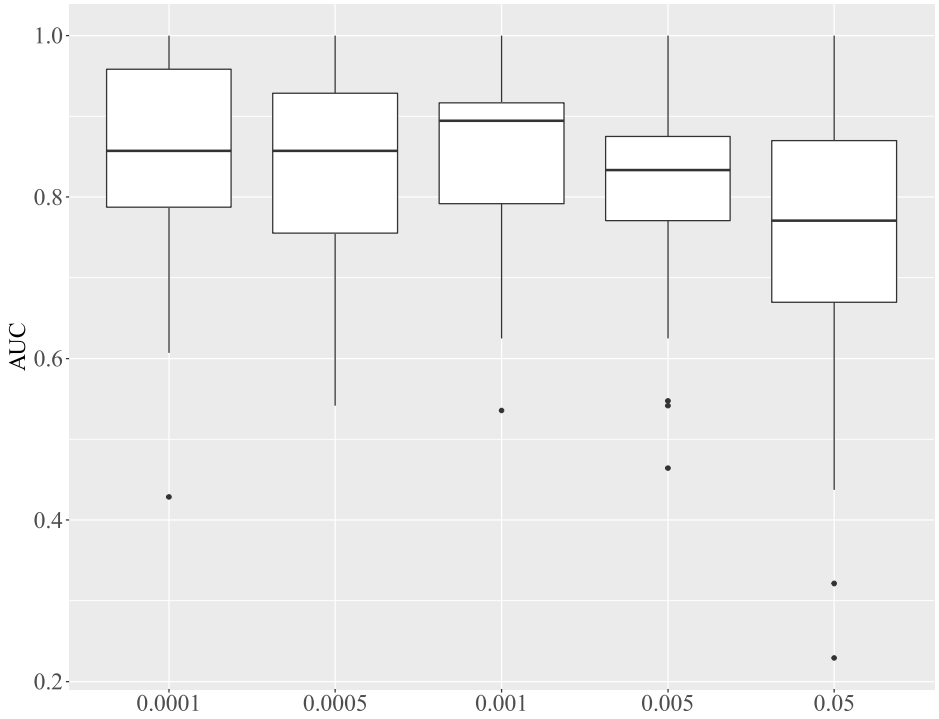


Figure 3.19. Average AUC performance under different learning rates of linear model weights.

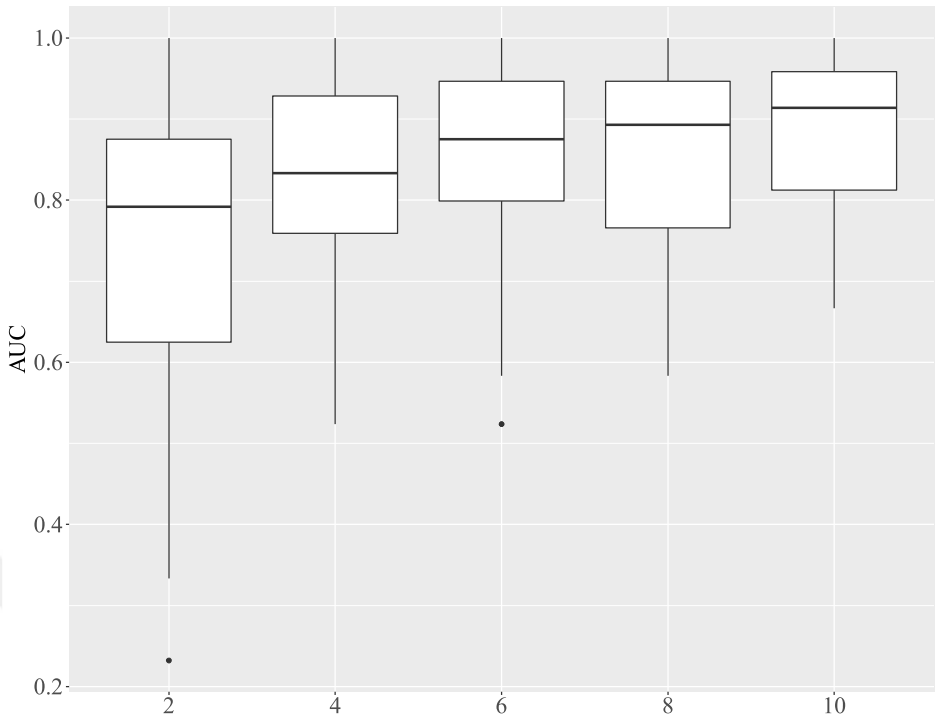


Figure 3.20. Average AUC performance under different number of prototypes.

3.9. Conclusion

This chapter presents a prototype learning framework for MIL problems. This simple architecture can be applied to data from all kinds of domains and offers interpretability of the solutions. Although the method is applied to the classical MIL datasets, its modification to different problems is an interesting research direction. Our aim is to present the simplicity of the approach. Therefore, logistic regression classifiers and Euclidean distance are adopted. We compared our results with well-known approaches in the field. Our approach is not the best performing approach among these state-of-art approaches. However, considering the flexibility of the architecture, one could incorporate more complex classifiers, different distance metrics and different aggregation procedures to obtain more powerful models. We applied our approach to a well-known MNIST problem in order to show interpretability power of our approach. As a result of this example we can learn perfectly interpretable prototypes.

4. PROTOTYPE FOREST IN MULTIPLE INSTANCE LEARNING

As mentioned in Section 2.2, embedding approaches are developed to overcome the problems of instance and bag-level approaches in MIL domain. The main disadvantage of instance level approaches are dimensionality and unavailability of instance-level labels. On the contrary, bag level approaches has a disadvantage of missing valuable instance level information. Even the main target of embedding approaches is to overcome disadvantages of instance level and bag level approaches, embedding approaches face certain challenges including computational cost of the algorithm (e.g. dimensionality), efficient parameter tuning, and loss of information (i.e. poor prediction performance).

To avoid the aforementioned problems with the existing approaches, this chapter proposes two new learning frameworks. Both learning frameworks utilize tree-based ensembles. The first approach constructs decision trees on the dissimilarity of the bags to randomly selected prototypes (i.e. bag-to-prototype distances) at each splitting step in decision tree learning. In this approach, each bag is represented by its dissimilarities to the randomly selected instances (i.e. prototypes) for embedding purposes. To be more specific, our proposal constructs decision trees on the dissimilarity of the bags to randomly selected prototypes (i.e. bag-to-prototype distances) at each splitting step in decision tree learning. A decision tree is a supervised learning approach that constructs tree-like learning algorithms. If the predicted target values are continuous, regression trees are constructed. However, if the target value is discrete, classification trees are constructed. Due to the randomness associated with the prototype selection, a tree-based ensemble learning framework is devised to classify the bags with the proposed bag-to-prototype distance representation. The method is called Prototype Forest (PF).

The second approach integrates PF and PL. The main motivation of the integration of two models is the random prototype selection of PF considering run time issues. The selected instances might be irrelevant from the bags' labels due to MIL assump-

tions. This may result in deeper trees that can increase run time. On the contrary, PL lead the algorithm create better splits. Three variations of the approach are proposed. The first variation, namely Tree-based Prototype Learning (TPL), constructs a single decision tree with the tuned hyperparameters of PL. TPL uses all bags in the fold instead of a stratified sampling of the bags. Therefore, a single tree with tuned parameters is expected to provide comparable performance. The second approach, namely Prototype Forest with Prototype Learning (PFPL), is similar to PF. A tree-based ensemble is built with a randomly selected hyperparameters of PL. The third variation, namely Linear Tree (LT), create splits using logistic regression outputs. LT is a linear approximation and utilizes a linear model output instead of a constant approximation. As a result, the approach is expected to increase the algorithm performance when compared to other variations.

MIL problems may suffer from the curse of dimensionality due to a large number of features. Therefore, principal component analysis (PCA) [91] is also applied for dimensionality reduction. The main motivation of dimensionality reduction is to remove the noise of the data and to decrease the run time of the algorithm [92]. This is also consistent with the MIL domain by definition. In MIL domain, certain dimensions might play an important role in labeling the bags whereas the remaining dimensions might have zero or very small impact on the labels. Compared to the existing dissimilarity-based approaches, our proposals scale well with the help of the hierarchical structure of the trees without sacrificing the prediction performance. Moreover, the idea of bootstrap aggregating as in Random Forests [60] avoids the problems with the overfitting and complex parameter tuning process.

4.1. Background

A simple classification example can be found in Table 4.1. In this example, there are four instances and each instance has three features and belongs to a class. The problem is a binary classification problem. Therefore, the class of each instance is either 0 or 1.

Table 4.1. A classification example.

Feature 1	**Feature 2**	**Feature 3**	**Class**
a	a	b	1
c	a	b	1
c	a	c	0
a	c	a	1

Two different decision trees for classification problem in Table 4.1 can be found in Figure 4.1 and Figure 4.2. Classification of the simple problem can be done by following the rules in both figures. Basically, Figure 4.1 explains that, if Feature 1 of an instance is "a", then its class or label is 1. If Feature 1 is "c", another internal node is created for Feature 3. If Feature 3 is "b", then class is 1. If Feature 3 is "c", class is 0.

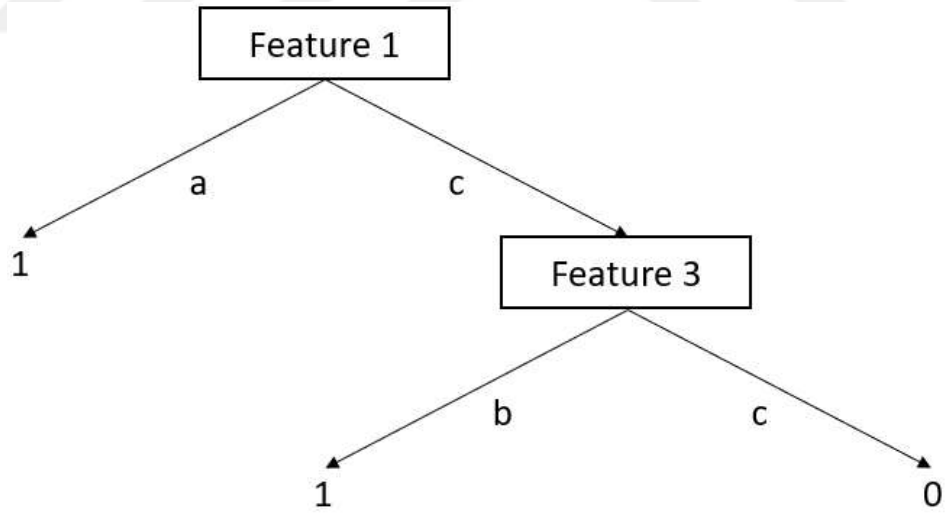


Figure 4.1. Classification decision tree 1 for simple problem.

Tree-based ensemble is a very powerful supervised learning method in that decision trees behave as an ensemble [60, 93]. To understand the logic of tree-based ensembles, an extended version of problem in Table 4.1 is created in Table 4.2. In this example, the number of features is increased from three to N. A decision tree similar to Figure 4.1 or Figure 4.2 can be constructed to this problem. However, decision trees are prone to overfitting and tree-based ensembles offer a solution to this problem [94].

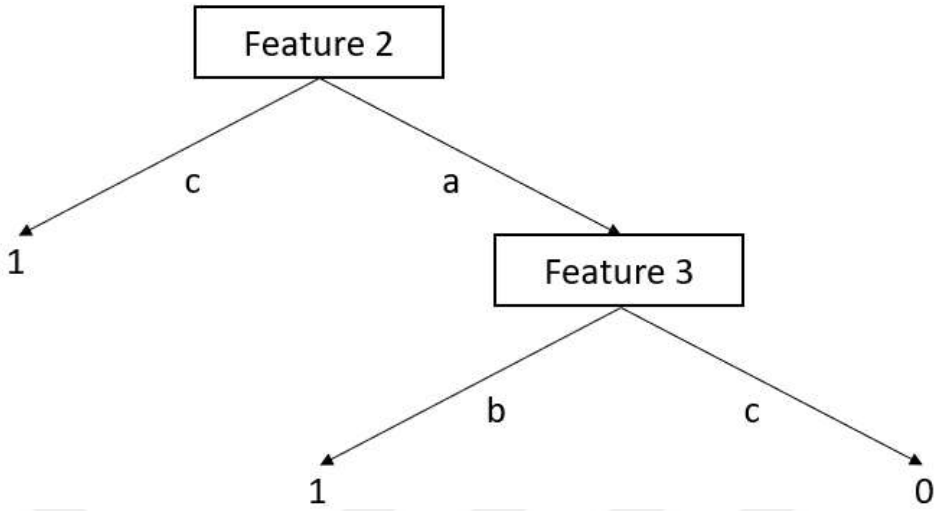


Figure 4.2. Classification decision tree 2 for simple problem.

Table 4.2. An extended classification example.

Feature 1	**Feature 2**	**Feature 3**	**...**	**Feature N**	**Class**
a	a	b	...	b	1
c	a	b	...	c	1
c	a	c	...	a	0
a	c	a	...	a	1

A basic overview to tree-based ensemble approach can be found in Figure 4.3. In tree-based ensemble algorithms, features are selected randomly in the first step. Then, a decision tree is constructed with the selected features. For example, if randomly selected features are Feature 2 and Feature 3, constructed decision tree is similar to Figure 4.2. Several decision trees are constructed in tree-based ensemble algorithms. In final step, majority vote of constructed trees predicts the target.

Tree-based ensembles outperform other state-of-art learning algorithms especially in high dimensional data [95] in terms of performance and running time of training and evaluation [96]. These advantages are also a motivation for this chapter.

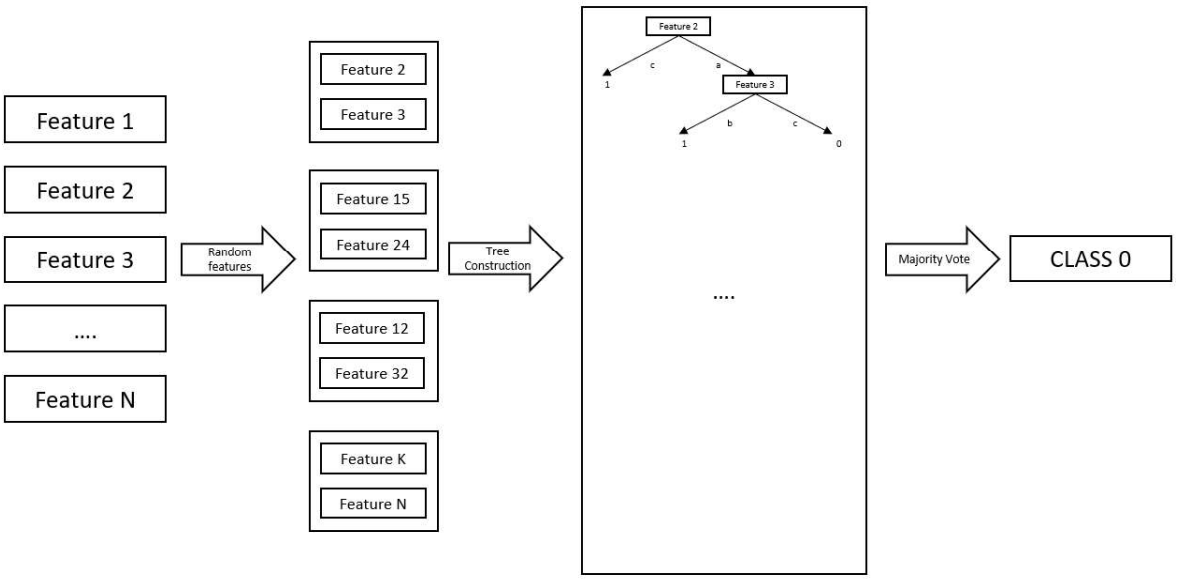


Figure 4.3. Tree-based ensemble structure.

4.2. Prototype Forest

Prototype Forest (PF) is a tree-based ensemble learning approach, which consists of τ trees. Figure 4.6 provides the details of the base algorithm, namely Prototype Tree. Each tree is built using a random selection of N bags out of M and a random selection of D prototypes is evaluated at each splitting decision. The stratified random sampling is performed to include bags from all classes. At each node of the tree, Step 8 calculates the instance to prototype distances for each bag. Then, the distances are aggregated at bag-level using the predefined bag to prototype dissimilarity features, namely minimum, maximum, and average. This results in N by $3D$ feature matrix for all selected bags. Step 9 evaluates each distance feature and split value combination based on Information Gain as in [97]. Entropy is a measure of impurity which is calculated as

$$E(S) = \sum_{i} -p_i \log_2 p_i. \tag{4.1}$$

The entropy of a pure node is zero, whereas the entropy of one indicates that a node is divided equally. Information Gain is the difference in entropy before and after

creating a new split. If Entropy decreases, Information Gain becomes positive. The split that gives the largest Information Gain becomes the next decision node. The splitting procedure is repeated at each child node until the node is homogeneous in class distribution or maximum tree depth (h) is achieved.

Figure 4.4 shows the procedure behind splitting during decision tree construction. This is a recursive procedure that stops when there is no Information Gain or the maximum depth of a tree is reached. The architecture is drawn with one bag to keep the simplicity. At the first step, all instances are combined as it is shown in Figure 4.4. After that, the prototype definition step is followed. Again, to keep the simplicity, random instances can be selected as prototypes. When the prototypes are determined, instance-to-prototype distances are calculated. Then these distances are aggregated at bag level using a pooling function such as minimum, maximum or average. An $M \times D$ matrix is created at the end of this aggregation. Then the parent node is split into child nodes using these distances.

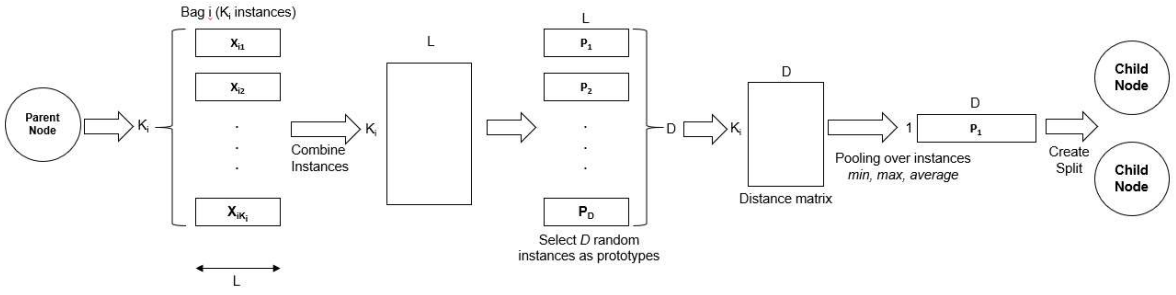


Figure 4.4. Overview of PF architecture. Bags are recursively partitioned based on their distances to randomly selected prototypes.

To illustrate the approach an example of 4 bags is created in Table 4.3. There are 4 bags and each bag has 2 instances and each instance has two features. A prototype of 1, 1 is selected for simplicity. Bag to prototype distances can be found in Table 4.4. When the algorithm is run on the data set, the best split value is 1.2 and the best feature is minimum (F_{min}). Therefore, the bags are splitted considerig the minimum distance less than 1.2 and more than 1.2 to the prototype. Figure 4.1 shows the entropies of root node and child nodes. There are two bags that are labelled as 1 and two bags

that are labeled as zero in the root node. Therefore, the initial entropy of 4 bags is 1. After splitting, there are two homogenous child nodes. Both nodes have 0 entropy.

Table 4.3. MIL example.

Label	**Bag Id**	**Instance Id**	**Feature 1**	**Feature 2**
1	1	1	1	1
1	1	2	0	1
1	2	1	0	1
1	2	2	2	4
0	3	1	2	1
0	3	2	3	2
0	4	1	2	3
0	4	2	1	3

Table 4.4. Bag to prototype distances.

	Bag 1	Bag 2	Bag 3	Bag 4
Max	1.00	3.16	2.23	2.23
Min	0.00	1.00	1.41	2.00
Mean	0.50	2.08	1.83	2.12

Prototype Tree algorithm is run on well known Musk 2 [19] data set to illustrate the approach. A simple decision tree example of the learning algorithm for Musk 2 data set is illustrated in Figure 4.7. The first line of each node in the figure shows the overall probability of having a 0 label. At the root node, 50% of all bags are labeled as 0. The second line shows the amount of bags at each node. As there is no splitting at the root node yet, 100% of all bags are presented in the root node. The number of prototypes is 1. Let $P_{1,1}$ be the prototype that is generated for the root node of the first tree. The root node of the first tree is split on the feature of minimum distance. As indicated from Figure 4.7, if $F_{min}(B_i, P_{1,1})$ is greater than 22, bag is labeled as 0.

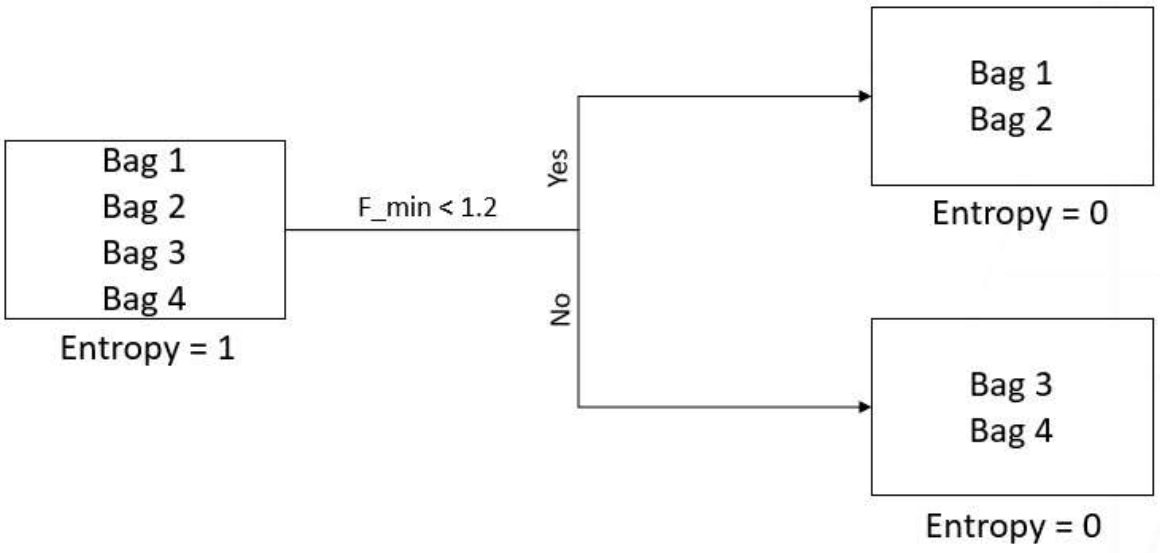


Figure 4.5. Entropy before and after the split.

Selection of the informative prototypes at Step 8 is in the core of the learning process. This requires a search for D prototypes in L dimensional space. A prototype learning approach, namely PL, is proposed by [17] but the approach is not computationally efficient due to hyperparameter tuning requirements. An alternative approach is the use of instances as prototype candidates. However, this choice can be computationally inefficient depending on the problem characteristics (i.e. the number of bags M and the number of instances in each bag K_i). To resolve such problems, we propose a random selection of candidate prototypes in an ensemble learning framework, namely PF. With the randomization of the selected prototypes combined with the ensemble learning strategy, the variance of the individual trees is reduced which improves the generalization behavior.

On the other hand, learning prototypes instead of random selection can provide benefits to the ensemble such as decreasing the depth of a tree due to obtaining the pure nodes faster. Moreover, the number of trees can be decreased with the additional cost of prototype learning. Therefore, we also consider a variation of prototype tree called TPL in which the selection of prototypes and the hyperparameter tuning are done with PL [17]. Due to greedy nature of decision trees, we also proposed an ensemble version of TPL, namely PFPL. Hyperparameters of PFPL are selected randomly from

a parameter set for cost reduction purposes. Details of the learning integration are found in Section 4.3.

Definitions:
$tree(\hat{\chi}, depth, D)$, where $\hat{\chi} = \{B_{\hat{i}} : \hat{i} = 1, \ldots, N\}$, is a subset of all bags $\chi = \{B_i : i = 1, \ldots, M\}$. D is the number of prototypes.
F is a feature matrix. $F = \{F_{min}, F_{max}, F_{mean}\}$ is a $N \times (3 * D)$ matrix with ijth entry is s_{ij}
if *cannot further split or depth* $= h$ **then**
    Designate this node as a terminal node
    Return
end
Select D prototypes, extract features F
Obtain p^*, the split point for column p
$\hat{\chi}_{left} \leftarrow$ bags (rows) i with $s_{ip} \leq p^*$
$\hat{\chi}_{right} \leftarrow$ bags (rows) i with $s_{ip} > p^*$
$depth = depth + 1$
$tree(\hat{\chi}_{left}, depth, D)$
$tree(\hat{\chi}_{right}, depth, D)$

Figure 4.6. Prototype tree algorithm.

There are three hyperparameters in the algorithm. These hyperparameters are the number of trees (τ), the number of prototypes (D), and the maximum depth of each decision tree (h). τ is the size of each forest. There are τ decision trees in each forest. At each node of each decision tree, D prototypes are generated. A feature matrix from the bag-to-prototype distances is created at each node. h limits the depth of each decision tree. Extreme values of h may lead the model to overfit or underfit.

Furthermore, PCA is also applied to data sets to remove the noise from the data and to decrease the run time of the algorithm. The sum of explained variance by each principal component is calculated and dimensionality reduction is done with respect to

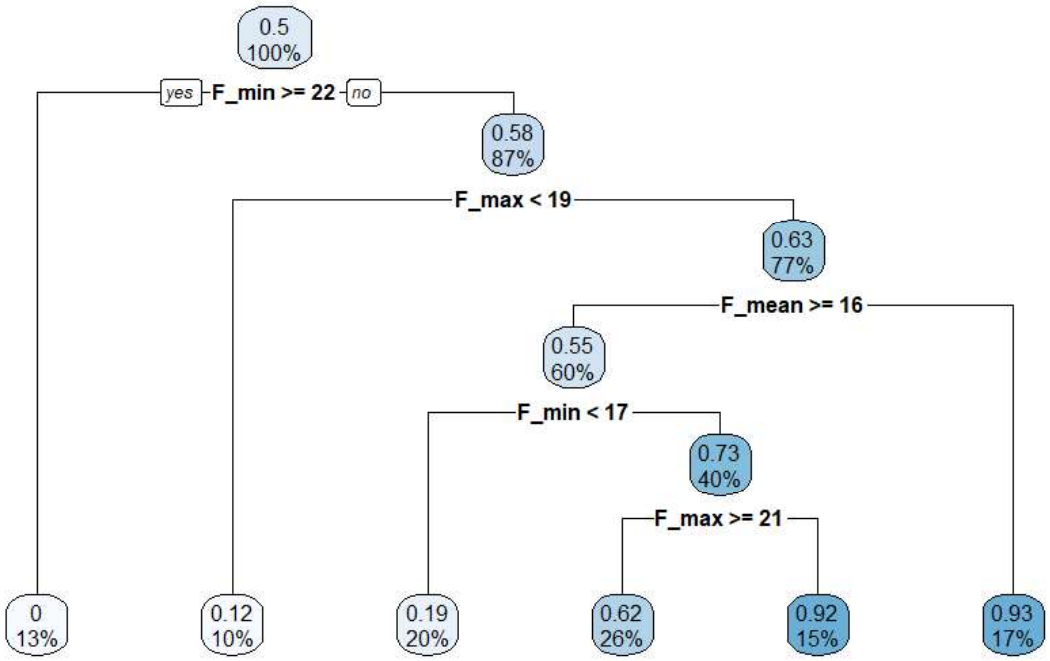


Figure 4.7. Prototype tree trained on Musk 2.

the cumulative total variance.

4.3. Prototype Forest with Prototype Learning

As mentioned before, at each step of PF, prototypes are selected randomly among the instances due to the run time considerations. As an extension, a method that learns prototypes instead of random selection might improve the algorithm's performance. Chapter 3 proposes a stochastic gradient descent algorithm that jointly learns prototypes and a linear classifier for MIL problems. The model learns logistic regression coefficients that predict the bags' labels. The well-known sigmoid function which is expressed as

$$\hat{y}_i = \beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id} \tag{4.2}$$

is used for binary classification. $\hat{y}$, the estimated probabilities for positive class are defined in Equation (4.2). The proposed model minimizes the cross-entropy loss which

is defined as

$$\mathcal{L}_{ce}(y, \hat{y}) = -yln(\sigma(\hat{y})) - (1-y)ln(1-\sigma(\hat{y})) \tag{4.3}$$

where y is the true label of a bag i and $\hat{y}$ is the estimation.

The objective function of the model can be formulated as

$$\min_{P,\beta} \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i, \hat{y}_i) + \lambda_w \left\|\beta\right\|_1 - \lambda_p \sum_{d=1}^{D}\sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{M}\sum_{d=1}^{D} \phi_{id}. \tag{4.4}$$

Prototypes, P, and classifier weights, β are the decision variables of the model. The classifier weights, the prototypes and the distances between the prototypes are regularized. λ_w, λ_p and λ_d are the related regularization parameters. The model constraints are

$$\Phi_{id}^{min} = \min_{X_{ij} \in B_i} Dist(X_{ij}, P_d) \tag{4.5}$$

$$\hat{y}_i = \sigma(\beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id}) \tag{4.6}$$

where Φ_{id}^{min} is the minimum distance between bag i and prototype d and $\hat{y}_i$ is the estimated label of bag i. Any differentiable function including minimum, maximum, or average can be used to define these distances. Equation (4.5) is the formulation of minimum distance. Other functions (e.g. maximum and average) can also be defined similarly.

The definitions of maximum and average distances are

$$\Phi_{id}^{max} = \max_{X_{ij} \in B_i} Dist(X_{ij}, P_d) \tag{4.7}$$

$$\Phi_{id}^{mean} = \frac{1}{K_i} \sum_{j=1}^{K_i} Dist(X_{ij}, P_d) \tag{4.8}$$

respectively. Both equations can be replaced with Equation (4.5). Maximum distance

is the maximum of pairwise distances between all instances to prototype d and the average distance is the average of instance to prototype pairwise distances.

TPL algorithm builds a decision tree with the tuned hyperparameters. The details of the base algorithm can be found in Figure 4.8. The difference of Figure 4.6 and Figure 4.8 is the Step 8. TPL learns prototypes instead of randomly selected prototypes. There are three hyperparameters in the PL algorithm. These hyperparameters are the learning rate of prototypes α_p, the learning rate of the classifier weights α_w, and the number of epochs. These hyperparameters are tuned as in [17] in TPL.

Definitions:
$tree(\hat{\chi}, depth, D)$, where $\hat{\chi} = \{B_{\hat{i}} : \hat{i} = 1, \ldots, N\}$, is a subset of all bags $\chi = \{B_i : i = 1, \ldots, M\}$. D is the number of prototypes.
F is a feature matrix. $F = \{F_{min}, F_{max}, F_{mean}\}$ is a $N \times (3 * D)$ matrix with ijth entry is s_{ij}
if *cannot further split or depth* $= h$ **then**
    Designate this node as a terminal node
    Return
end
Learn D prototypes with tuned hyperparameters, extract features F
Obtain p^*, the split point for column p
$\hat{\chi}_{left} \leftarrow$ bags (rows) i with $s_{ip} \leq p^*$
$\hat{\chi}_{right} \leftarrow$ bags (rows) i with $s_{ip} > p^*$
$depth = depth + 1$
$tree(\hat{\chi}_{left}, depth, D)$
$tree(\hat{\chi}_{right}, depth, D)$

Figure 4.8. TPL algorithm.

The details of the base algorithm of PFPL can be found in Figure 4.9. Similarly, the difference between PFPL and PF is Step 8. In PFPL, the hyperparameters are selected randomly among a configuration set. Figure 4.9 shows the generation of a

decision tree. A tree-based ensemble is built several trees in PFPL. On the contrary, TPL builds a single decision tree.

Definitions:
$tree(\hat{\chi}, depth, D)$, where $\hat{\chi} = \{B_{\hat{i}} : \hat{i} = 1, \ldots, N\}$, is a subset of all bags $\chi = \{B_i : i = 1, \ldots, M\}$. D is the number of prototypes.
F is a feature matrix. $F = \{F_{min}, F_{max}, F_{mean}\}$ is a $N \times (3 * D)$ matrix with $\hat{i}j$th entry is $s_{\hat{i}j}$
if *cannot further split or depth* $= h$ **then**
    Designate this node as a terminal node
    Return
end
Learn D prototypes with randomly selected hyperparameters, extract features F
Obtain p^*, the split point for column p
$\hat{\chi}_{left} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}p} \leq p^*$
$\hat{\chi}_{right} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}p} > p^*$
$depth = depth + 1$
$tree(\hat{\chi}_{left}, depth, D)$
$tree(\hat{\chi}_{right}, depth, D)$

Figure 4.9. PFPL algorithm.

4.4. Linear Tree in Prototype Forest

Linear model trees learn a linear hyperplane as opposed to the axis parallel splits of traditional decision trees [98]. Figure 4.10 explains the difference between the model trees and traditional decision trees. There are instances from three different classes. The behavior of the data requires a better splitting mechanism other than axis parallel splitting. Applying linear tree classifier to the problem provides better classification results when compared to the traditional decision tree classifier.

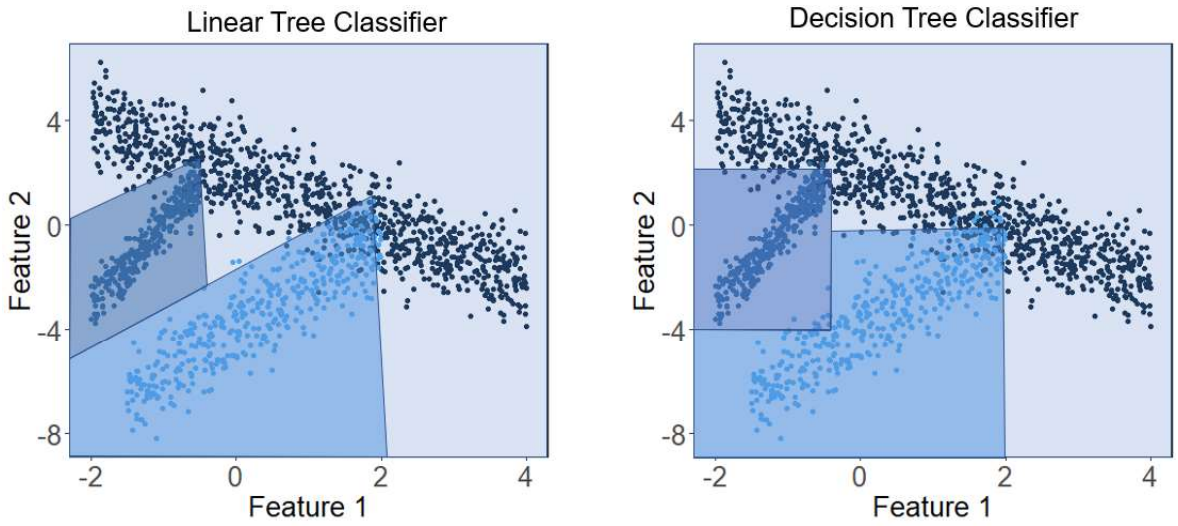


Figure 4.10. Linear model tree vs decision tree.

The base algorithm details of Linear Tree (LT) can be found in Figure 4.11. The steps until Step 8 are similar to the base algorithm of PFPL. However, instead of extracting distance features and splitting the bags considering distance features, logistic regression is fit and labels of each bag, namely $\hat{y}$, are estimated. The bags are split regarding with the estimated bag labels.

4.5. Time Complexity

The time complexity of constructing a decision tree is $\mathcal{O}(n \log n)$ where n is the number of rows. In Figure 4.6, n is the number of bags, i.e. M. Therefore, the complexity of constructing a decision tree becomes $\mathcal{O}(M \log M)$. As the maximum depth of each tree is capped with a constant h, complexity turns to be $\mathcal{O}(Mh)$. All features, F, are needed to be tested in each split of the tree. In addition to that, the creation of a feature matrix adds a complexity of K, i.e. maximum number of instances, and D, i.e. number of prototypes. Finally, the time complexity of constructing one decision tree would be $\mathcal{O}(FKDMh)$ in Figure 4.6. This iteration will be repeated for τ times while building a forest. The time complexity of PF is $\mathcal{O}(\tau FKDMh)$. The approach has a linear time complexity depending on the problem and algorithm parameters.

Definitions:

$tree(\hat{\chi}, depth, D)$, where $\hat{\chi} = \{B_{\hat{i}} : \hat{i} = 1, \ldots, N\}$, is a subset of all bags $\chi = \{B_i : i = 1, \ldots, M\}$. D is the number of prototypes.

F is a feature matrix. $F = \{\hat{y}_i\}$ is a $N \times 1$ matrix with $\hat{i}$th entry is $s_{\hat{i}}$

if *cannot further split or depth* $= h$ **then**

    Designate this node as a terminal node

    Return

end

Learn D prototypes, extract features F

Obtain p^*, the split point

$\hat{\chi}_{left} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}} \leq p^*$

$\hat{\chi}_{right} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}} > p^*$

$depth = depth + 1$

$tree(\hat{\chi}_{left}, depth, D)$

$tree(\hat{\chi}_{right}, depth, D)$

Figure 4.11. Linear tree in prototype tree algorithm.

The time complexity of PFPL algorithm is as follows. Let E be the number of epochs and N be the number of points in the data set. Distance calculation, aggregation of features, and classifier calculation have the complexity of $\mathcal{O}(DKL + DK + D)$. The complexity can be simplified to $\mathcal{O}(DKL)$. Therefore, the prototype learning procedure of PFPL adds linear $\mathcal{O}(ENDKL)$ complexity at each node.

4.6. PCA Analysis of Data Sets

To understand the performance of PF deeper, we analyzed results of selected data sets. Furthermore, we presented the effect of PCA on the performance of the algorithm. First, the average distances of bags and prototypes are calculated. As the prototypes are selected among the instances, the average minimum, maximum and mean distances for all instance to bag pairs are considered. We selected two data sets, Web 8 which

has a lower AUC value without PCA (0.658) and Hammond Flycatcher which has the highest AUC value for PF. Figure 4.12 shows the box plot of the average distances of prototypes that are selected from class 0 bags to bags of class 0 and 1. Similarly, Figure 4.13 shows the box plot of the average distances of prototypes that are selected from class 1 bags to bags of class 0 and 1. Same plots for Hammonds Flycatcher data set can be found in Figure 4.14 and Figure 4.15 respectively. As figures show, for Web 8 data set, average of minimum, maximum and mean distances to 0 and 1 labeled bags do not differ. However, average of both minimum and mean distances to 0 and 1 labeled bags differ for Hammonds Flycatcher data set.

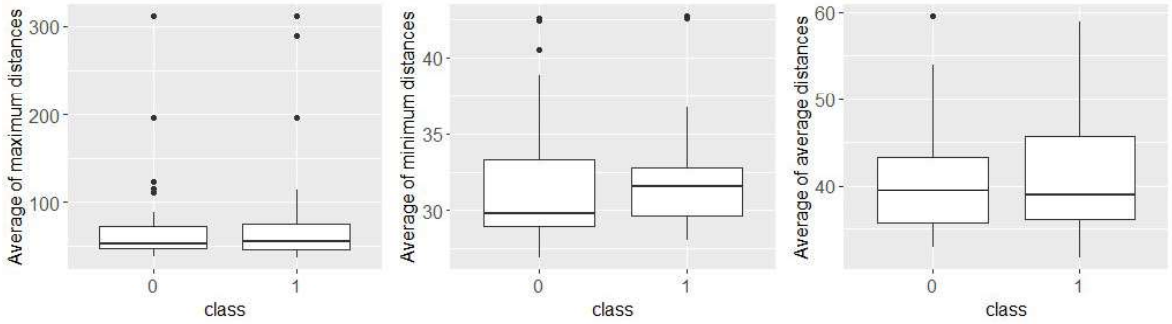


Figure 4.12. Average distances between prototypes and bags when Web 8 instances are selected from bags of class 0.

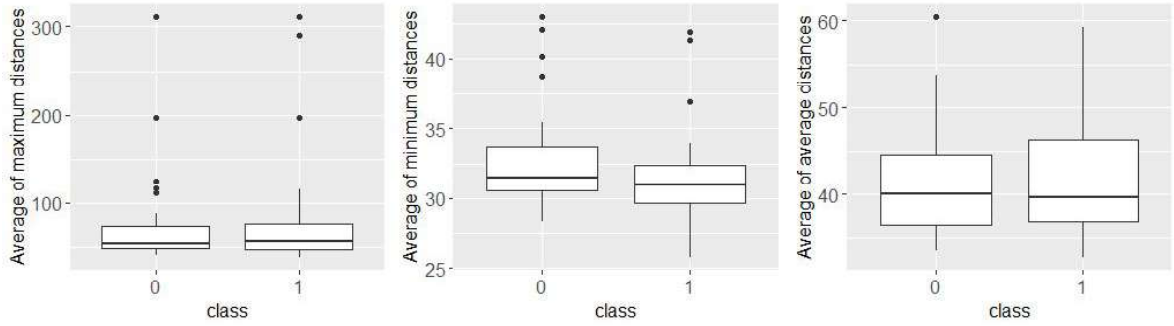


Figure 4.13. Average distances between prototypes and bags when Web 8 instances are selected from bags of class 1.

We continue with a statistical analysis to understand whether this interpretation is valid or not. We applied Mann-Whitney U Test [99] to compare the average of distances to different labeled bags. This test can be considered as the nonparametric ver-

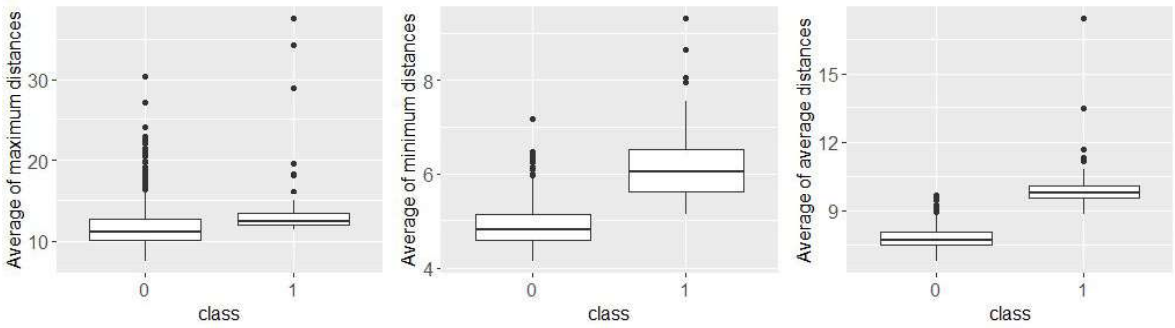


Figure 4.14. Average distances between prototypes and bags when Hammonds Flycatcher instances are selected from bags of class 0.

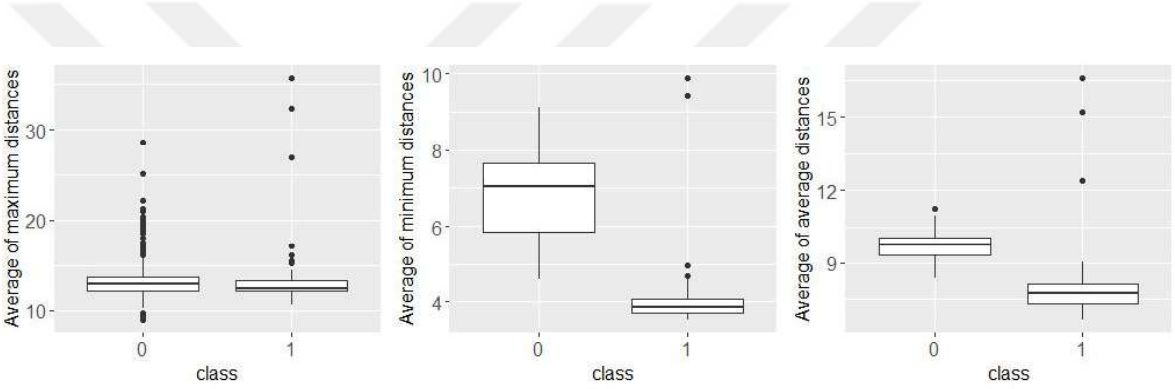


Figure 4.15. Average distances between prototypes and bags when Hammonds Flycatcher instances are selected from bags of class 1.

sion of the parametric t-test. The main reason behind the selection of Mann-Whitney U Test is that, the metrics do not meet the distribution assumption of the t-test. For the average of maximum distances, the null hypothesis states that the distances of prototypes to 0 and 1 labeled bags belong to identical populations. We applied Mann-Whitney U Test to compare the independent samples to test this hypothesis. p-values of all tests can be found in Table 4.5. As p-values show, we fail to reject null hypothesis at significance level $\alpha = 0.05$ for all three features for Web 8 data set. That indicates, prototypes (instances in our experiments) to bags distances are not significantly different for 0 and 1 labeled bags. However, we reject null hypothesis when we repeat the tests for Hammonds Flycatcher data set. We can conclude that bag to prototype distances are significantly different from each other for all features for Hammonds Flycatcher.

Table 4.5. Mann-Whitney U test results.

Data set	**Distance Type**	**Prototype Class**	p**-value**
Web8	Min	0	0.2587
Web8	Max	0	0.7294
Web8	Mean	0	0.7873
Web8	Min	1	0.0781
Web8	Max	1	0.6820
Web8	Mean	1	0.8864
HFlycatch.	Min	0	0.0000
HFlycatch.	Max	0	0.0000
HFlycatch.	Mean	0	0.0000
HFlycatch.	Min	1	0.0000
HFlycatch.	Max	1	0.0350
HFlycatch.	Mean	1	0.0000

To illustrate these analyses, sample bags and a sample instance are extracted. 7 different sample bags from Web 8 data set are shown in different colors in Figure 4.16 and classes of these bags are shown in Figure 4.17. PF selects random instances among all training bags. A random instance can be seen in Figure 4.18. This single instance has the Web data set characteristics. Instances of Web data sets have a few positive values and the rest of the features have small deviations around 0. These positive values accumulate on certain features according to the bag that instance belongs to. In other words, classification should be designed regarding the order of the feature rather than its value for Web data sets. Instance-224 from the Figure 4.18 belongs to Bag-11. Therefore, this instance has similar pairwise distances to all bags other than Bag-11, independent from the class. The reason behind this is, Instance-224 has positive values for the features that the instances from all other bags have small deviations around 0 and Instance-224 has almost 0 values for the features that the instances from other bags have positive values.

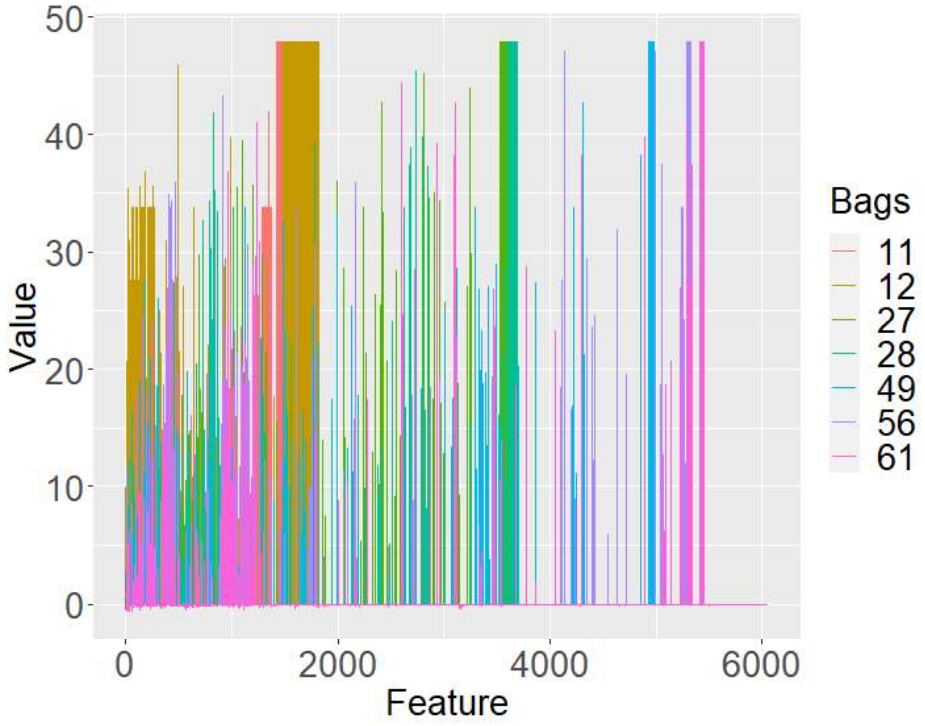


Figure 4.16. Sample bags by bag id.

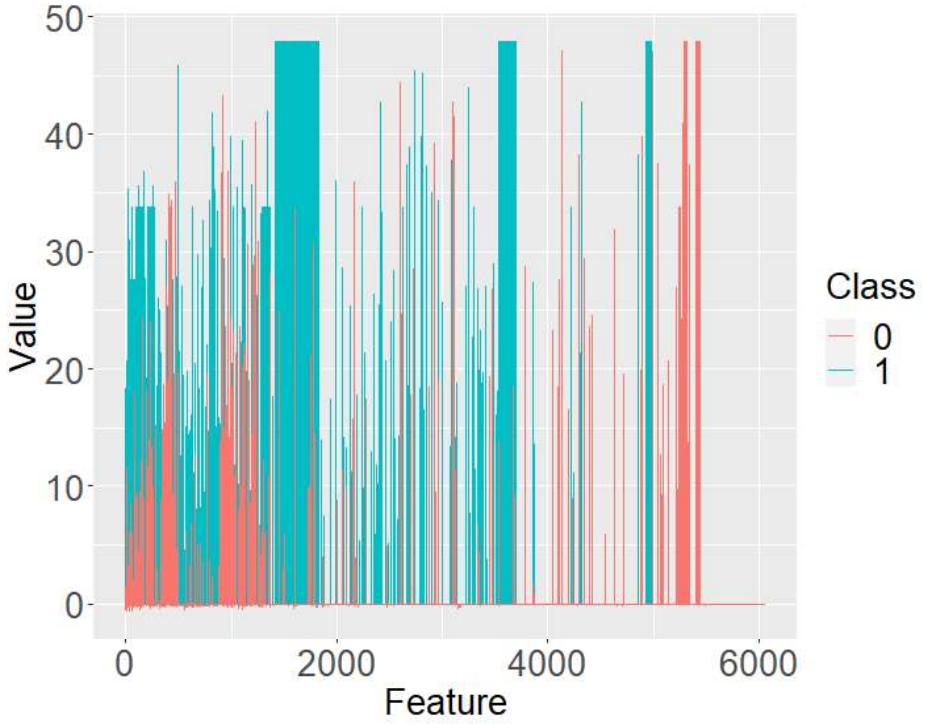


Figure 4.17. Sample bags by class.

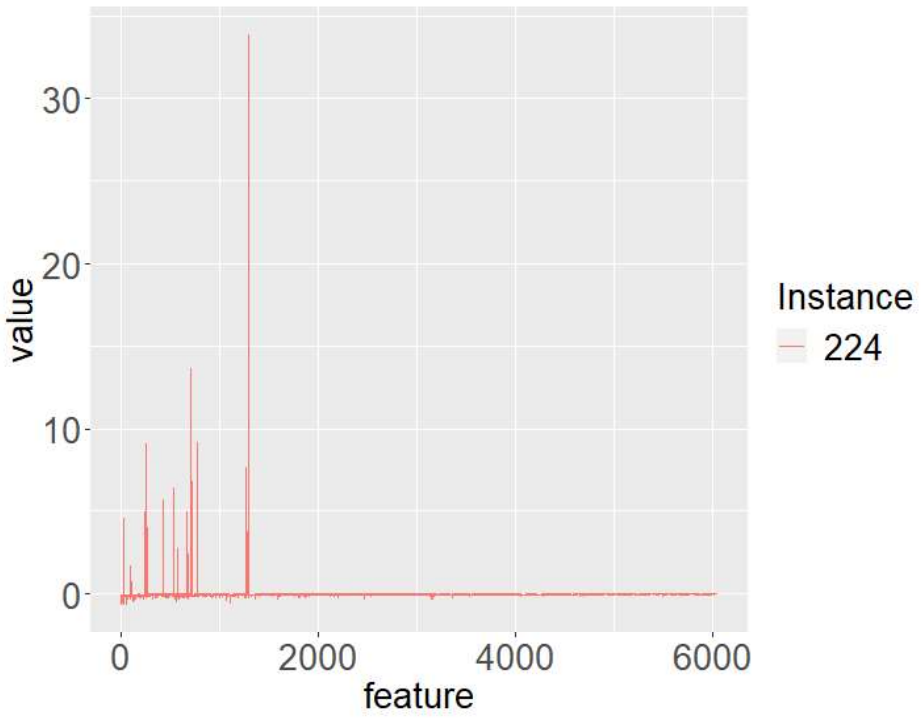


Figure 4.18. Sample instance from Web 8.

These results explain why the PF's outcomes for Web 8 and Hammonds Flycatcher data sets differ in terms of AUC value. In order to improve the algorithm's performance, principal component analysis (PCA) is applied to data sets. PCA [91] is a method that identifies the directions where the data has the biggest variance. PCA can be used to reduce the dimensions of the data sets. Figure 4.19 shows the amount of variance explained by the first 20 principal components for Web 8 data set. As indicated in the Figure 4.19, 52% of total variance can be explained by the first 20 principal components even the data set has 5999 dimensions. After the 20^{th}, each principal component explains less than 0.6% of the total variance.

For that purpose, PCA is added as a hyperparameter to the PF algorithm. Figure 4.20 shows the explained variance of the principal components. The main aim is to map the data into a lower dimensional space using this information. After adding PCA to PF, the performance of Web 8 data set increased to 0.938.

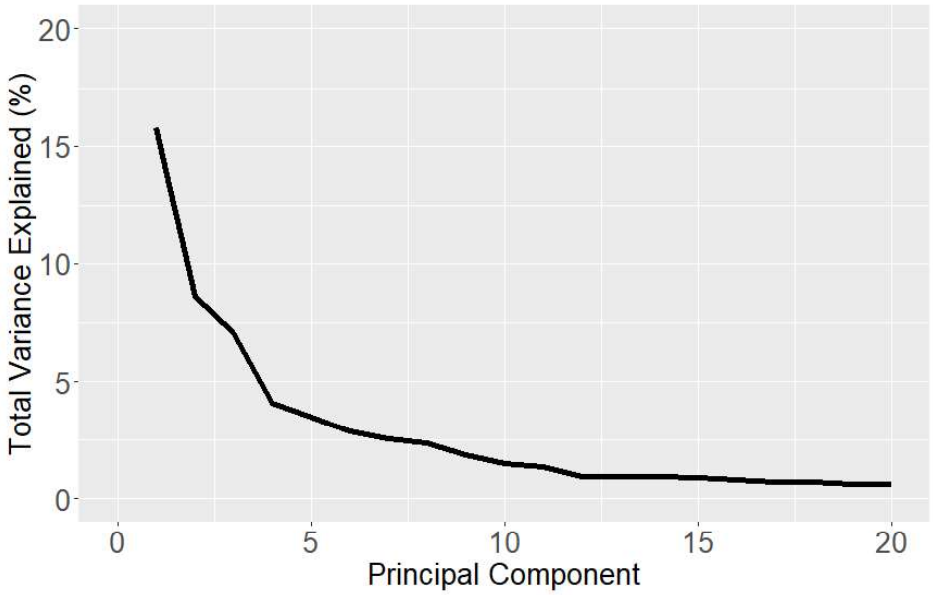


Figure 4.19. Total variance explained by the first 20 principal components.

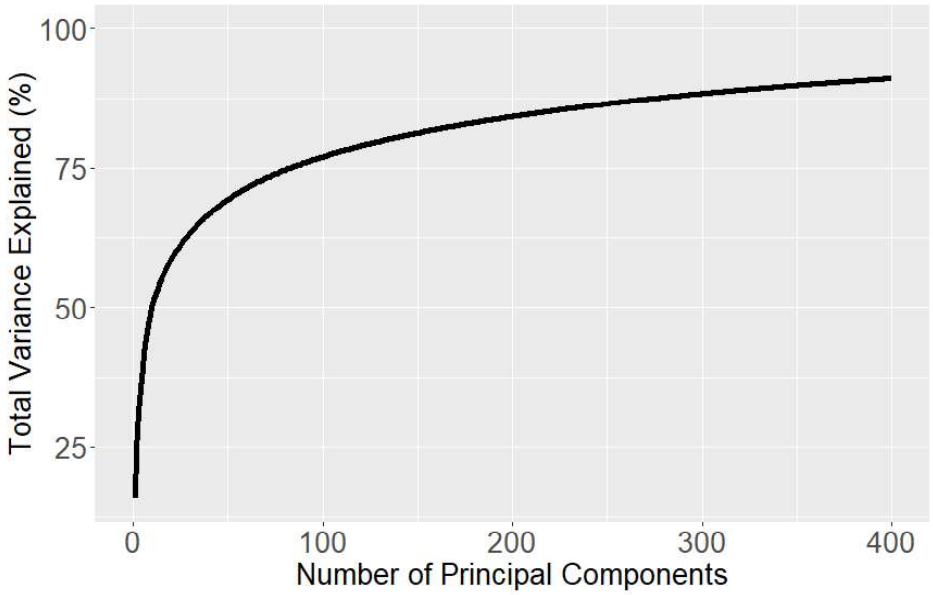


Figure 4.20. Explained variance by principal components for Web 8.

4.7. Experimental Results

Each data set is run with stratified 10-fold cross-validation which is repeated five times. Cross-validation indices are randomly generated by [84]. Average AUC

scores of state-of-the-art MIL approaches and PF is reported. Considered well-known benchmarks are APR [19], CCE [85], Citation - KNN [7], D_{maxmin}, $D_{meanmin}$, D_{minmin} [44], MILES [2], miFV [86], QP-MIL [63], and PL.

A grid search is applied to tune the hyperparameters of the model. Out-of-bag prediction errors are used to select the best parameter combination for each fold. The tuned hyperparameters are the number of prototypes at each node, the maximum depth of the tree, the number of trees to be constructed, and cumulative total variance explained for each forest. These parameters are selected among {1, 2, 4, 8}, {4, 8, 16}, {50, 100, 200}, and {50%, 60%, 70%, 80%, 90%, 95%, NoPCA} respectively. Three distance features, *minimum*, *maximum*, and *mean* are used. Each tree in the ensemble is constructed with stratified sampled 70% of the training bags. The boundaries of regularization parameters and learning parameters of PL are $[0.00005, 0.005]$ and $[0.00001, 0.001]$ respectively. Learning rates were decreased 50% every 40 epochs. All experiments were completed on a Windows Server 2016 operating system. The installed memory of the system is 16.0 GB memory (RAM) and the system has a 2.30 GHz Intel Xeon(R) Processor.

Results of all PF, TPL, PFPL, and LT experiments, and the state-of-the-art approaches can be found in Table 4.6. Experiments show that all four methods shows a competitive performance among all data sets. Out of 71 data sets, PFPL ranks in the top five performing algorithms in 49 (69%) data sets. LT is in the top five performing approaches in 46 (65%) data sets. PF and TPL are the top five best performing models in 37 (52%) and 33 (46%) data sets respectively. PF's performance is worse than other methods in 2 data sets. In other words for 3% of the data sets, PFs rank is in bottom three among 14 approaches. PFPL, TPL, and LTs performances are not in bottom three in terms of average AUC.

Table 4.6. Results of the algorithms.

Avg AUC	**APR**	**CCE**	**Cit.**	**Dmx**	**Dmn**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
Musk1	0.772	0.882	0.871	0.920	0.945	0.931	0.896	0.874	**0.968**	0.909	0.946	0.935	0.942	0.944
Musk2	0.806	0.781	0.850	0.956	**0.976**	0.956	0.823	0.786	0.945	0.892	0.897	0.915	0.906	0.908
Mutagen1	0.501	0.833	0.827	0.820	0.851	0.766	0.910	0.909	0.855	0.788	0.910	0.910	**0.917**	0.914
Mutagen2	0.458	0.727	0.688	0.590	0.647	0.345	**0.882**	0.867	0.785	0.873	0.775	0.732	0.783	0.868
Protein	0.509	0.643	0.552	0.561	0.523	**0.876**	0.872	0.873	0.851	0.867	0.807	0.796	0.818	0.855
Elephant	0.728	0.854	0.882	0.876	0.936	0.915	0.886	0.882	**0.941**	0.936	0.876	0.845	0.885	0.915
Fox	0.585	0.623	0.569	0.457	0.612	0.704	0.630	0.655	0.677	**0.737**	0.695	0.632	0.712	**0.748**
Tiger	0.583	0.832	0.752	0.761	0.853	0.850	0.826	0.860	**0.901**	0.892	0.847	0.824	0.861	0.855
CrAfrican	0.580	0.840	0.862	0.939	**0.967**	0.966	0.836	0.851	0.953	0.902	0.948	0.930	0.947	0.952
CrAntique	0.595	0.722	0.747	0.877	**0.922**	0.910	0.769	0.801	0.871	0.894	0.901	0.892	0.915	0.894
CrBattle.	0.569	0.912	0.866	0.972	**0.981**	0.966	0.876	0.889	0.958	0.972	0.951	0.949	0.964	0.977
CrBeach	0.596	0.969	0.926	0.972	0.983	0.990	0.977	0.980	**0.993**	0.979	0.988	0.979	0.986	0.992
CrBuses	0.591	0.963	0.848	0.962	0.973	**0.981**	0.944	0.944	0.960	0.960	0.963	0.962	0.969	0.971
CrCars	0.608	0.878	0.835	0.930	**0.948**	0.916	0.856	0.864	0.912	0.905	0.911	0.920	0.931	0.944
CrDesserts	0.571	0.922	0.866	0.950	**0.974**	0.966	0.915	0.943	0.969	0.956	0.944	0.958	0.963	0.965
CrDin.	0.551	0.897	0.802	0.948	**0.983**	0.982	0.882	0.916	0.964	0.959	0.951	0.943	0.952	0.957
CrDogs	0.570	0.805	0.752	0.894	**0.919**	0.911	0.838	0.856	0.863	0.844	0.852	0.901	0.869	0.909

Table 4.6. Results of the algorithms. (cont.)

Data set	**APR**	**CCE**	**Cit.**	**Dmx**	**Dmn**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
CrElep.	0.643	0.893	0.870	0.966	**0.983**	0.970	0.858	0.884	0.968	0.923	0.943	0.951	0.950	0.955
CrFashion	0.868	0.950	0.909	0.983	0.990	0.991	0.904	0.928	0.986	0.953	0.982	0.991	**0.993**	0.990
CrFlowers	0.636	0.844	0.833	0.932	0.947	**0.953**	0.856	0.890	0.938	0.885	0.928	0.933	0.934	**0.957**
CrFood	0.860	0.985	0.934	0.993	**0.998**	0.996	0.979	0.984	0.992	0.983	0.986	0.983	0.988	0.991
CrHist.	0.773	0.978	0.942	0.984	**0.998**	0.994	0.985	0.987	0.994	0.929	0.987	0.989	0.991	0.997
CrHorses	0.613	0.809	0.746	0.861	**0.920**	0.917	0.798	0.815	0.860	0.859	0.896	0.876	0.901	0.868
CrLizards	0.553	0.938	0.897	0.960	**0.980**	0.971	0.940	0.934	0.973	0.953	0.947	0.954	0.956	0.960
CrMount.	0.981	0.988	0.979	0.998	**1.000**	0.999	0.988	0.993	0.998	0.963	0.999	0.999	0.999	**1.000**
CrSkiing	0.509	0.875	0.770	0.956	**0.960**	0.953	0.867	0.873	0.960	0.923	0.931	0.923	0.928	0.938
CrSunset	0.495	0.728	0.706	0.803	**0.837**	0.751	0.704	0.751	0.739	0.766	0.770	0.786	0.788	0.802
CrWater.	0.595	0.836	0.865	0.962	**0.975**	0.966	0.839	0.896	0.955	0.891	0.929	0.946	0.944	0.948
UCSBBr.	0.569	0.644	0.706	0.725	0.831	0.791	0.823	0.848	0.888	**0.908**	0.810	0.857	0.825	0.901
Ng1	0.500	0.792	0.803	0.905	**0.941**	0.500	0.667	0.654	0.933	0.837	0.893	0.887	0.908	0.866
Ng2	0.508	0.660	0.638	0.895	**0.898**	0.554	0.718	0.614	0.831	0.870	0.865	0.866	0.868	0.874
Ng3	0.500	0.622	0.584	**0.818**	0.810	0.500	0.650	0.660	0.778	0.763	0.795	0.763	0.803	0.769
Ng4	0.508	0.678	0.636	0.822	0.857	0.479	0.704	0.663	0.820	0.845	**0.873**	0.871	0.869	0.845
Ng5	0.488	0.609	0.585	**0.853**	0.852	0.559	0.649	0.638	0.812	0.822	0.837	0.796	0.841	0.810

Table 4.6. Results of the algorithms. (cont.)

Data set	**APR**	**CCE**	**Cit.**	**Dmx**	**Dmn**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
Ng6	0.500	0.742	0.732	0.865	**0.890**	0.572	0.676	0.650	0.849	0.882	0.829	0.770	0.836	0.805
Ng7	0.496	0.648	0.633	0.752	0.790	0.547	0.645	0.654	0.816	0.742	0.842	0.820	**0.845**	**0.851**
Ng8	0.498	0.692	0.570	0.840	**0.870**	0.460	0.713	0.623	0.818	0.764	0.815	0.850	0.826	0.864
Ng9	0.500	0.806	0.818	0.348	0.326	0.500	0.699	0.616	0.850	0.881	0.868	**0.898**	0.877	0.819
Ng10	0.500	0.747	0.821	**0.918**	0.914	0.476	0.624	0.652	0.880	0.747	0.867	0.854	0.878	**0.918**
Ng11	0.492	0.794	0.714	**0.968**	0.958	0.460	0.667	0.634	0.923	0.810	0.960	0.947	0.962	0.962
Ng12	0.510	0.756	0.806	**0.868**	0.840	0.466	0.634	0.642	0.865	0.786	0.804	0.823	0.822	0.846
Ng13	0.500	0.528	0.621	0.932	0.946	0.500	0.468	0.632	**0.949**	0.921	0.904	0.930	0.923	0.927
Ng14	0.500	0.777	0.780	0.922	**0.942**	0.465	0.666	0.641	0.888	0.728	0.886	0.887	0.885	0.937
Ng15	0.500	0.790	0.763	0.874	0.905	0.517	0.621	0.644	0.921	0.858	**0.929**	0.921	0.928	0.870
Ng16	0.500	0.787	0.773	0.879	**0.898**	0.509	0.640	0.627	0.822	0.882	0.859	0.821	0.870	0.891
Ng17	0.494	0.766	0.692	0.818	**0.874**	0.531	0.680	0.620	0.796	0.760	0.857	0.814	0.853	**0.899**
Ng18	0.500	0.825	0.845	0.833	**0.874**	0.463	0.688	0.666	0.852	0.793	0.857	0.861	0.860	0.842
Ng19	0.502	0.831	0.766	0.785	0.802	0.561	0.642	0.638	0.814	**0.856**	0.711	0.716	0.728	0.850
Ng20	0.498	0.770	0.646	**0.839**	0.834	0.443	0.625	0.642	0.813	0.726	0.751	0.806	0.768	0.805
Web1	0.547	0.788	0.614	0.411	0.634	0.788	0.826	0.834	0.655	0.829	0.841	0.796	**0.856**	0.821
Web2	0.522	0.473	0.445	0.501	0.474	0.521	**0.700**	0.693	0.537	0.691	0.648	0.615	0.644	0.646

Table 4.6. Results of the algorithms. (cont.)

Data set	**APR**	**CCE**	**Cit.**	**Dmx**	**Dmn**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
Web3	0.600	0.603	0.649	0.501	0.708	0.608	0.772	0.767	0.661	0.733	0.959	**0.990**	0.974	0.976
Web4	0.575	0.834	0.707	0.548	0.799	0.629	0.817	**0.847**	0.627	0.794	0.815	0.799	0.823	0.814
Web5	0.540	0.612	0.506	0.507	0.711	0.794	0.731	0.776	0.552	0.678	0.893	**0.898**	0.888	**0.899**
Web6	0.581	0.621	0.520	0.494	0.525	0.543	0.722	0.727	0.650	0.741	0.857	0.854	**0.877**	0.729
Web7	0.586	0.592	0.675	0.600	0.690	0.670	0.690	0.688	0.545	0.744	0.924	**0.934**	0.927	0.904
Web8	0.552	0.575	0.498	0.579	0.409	0.366	0.652	0.650	0.530	0.710	0.937	0.938	**0.947**	0.759
Web9	0.595	0.631	0.599	0.497	0.735	0.687	0.730	0.741	0.503	**0.767**	0.734	0.554	0.736	0.730
Brown.	0.592	0.945	0.883	0.729	0.899	0.927	0.989	0.992	0.991	0.926	**0.993**	0.989	0.988	0.973
Chestnut.	0.520	0.827	0.802	0.831	0.853	0.749	0.898	0.910	**0.917**	0.844	0.893	0.909	0.902	0.881
Dark-eyed.	0.683	0.667	0.594	0.695	0.856	0.870	0.938	0.947	0.932	0.786	0.942	0.944	**0.956**	0.906
Hammon.	0.534	0.992	0.884	0.718	0.944	0.883	0.999	0.999	**1.000**	0.996	**1.000**	**1.000**	**1.000**	0.997
HermitTh.	0.668	0.488	0.535	0.570	0.578	0.892	0.808	0.836	**0.903**	0.750	0.785	0.845	0.801	0.847
HermitWa.	0.593	0.818	0.697	0.735	0.781	0.926	0.981	0.979	**0.984**	0.880	0.965	0.978	**0.984**	0.954
Olive.	0.644	0.878	0.794	0.853	0.896	0.924	**0.969**	0.963	0.967	0.933	0.957	0.965	0.967	0.951
Pacific.	0.531	0.836	0.699	0.723	0.754	0.769	0.957	**0.958**	0.943	0.831	0.952	0.957	0.955	0.899
Red-br.	0.583	0.872	0.771	0.803	0.876	0.877	0.979	**0.982**	0.971	0.888	0.962	0.970	0.977	0.934
Swainsons.	0.519	0.826	0.687	0.782	0.767	0.884	0.968	0.981	0.977	0.771	0.971	0.983	**0.989**	0.929

Table 4.6. Results of the algorithms. (cont.)

Data set	**APR**	**CCE**	**Cit.**	**Dmx**	**Dmn**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
Varied.	0.584	0.869	0.782	0.751	0.840	0.940	**1.000**	**1.000**	0.997	0.949	0.994	0.998	0.999	0.980
Western.	0.696	0.878	0.755	0.475	0.849	0.824	**0.988**	0.982	0.973	0.897	0.967	0.979	0.981	0.935
WinterWr.	0.598	0.965	0.907	0.944	0.932	0.854	0.993	0.990	0.988	0.928	0.991	0.991	**0.995**	0.984

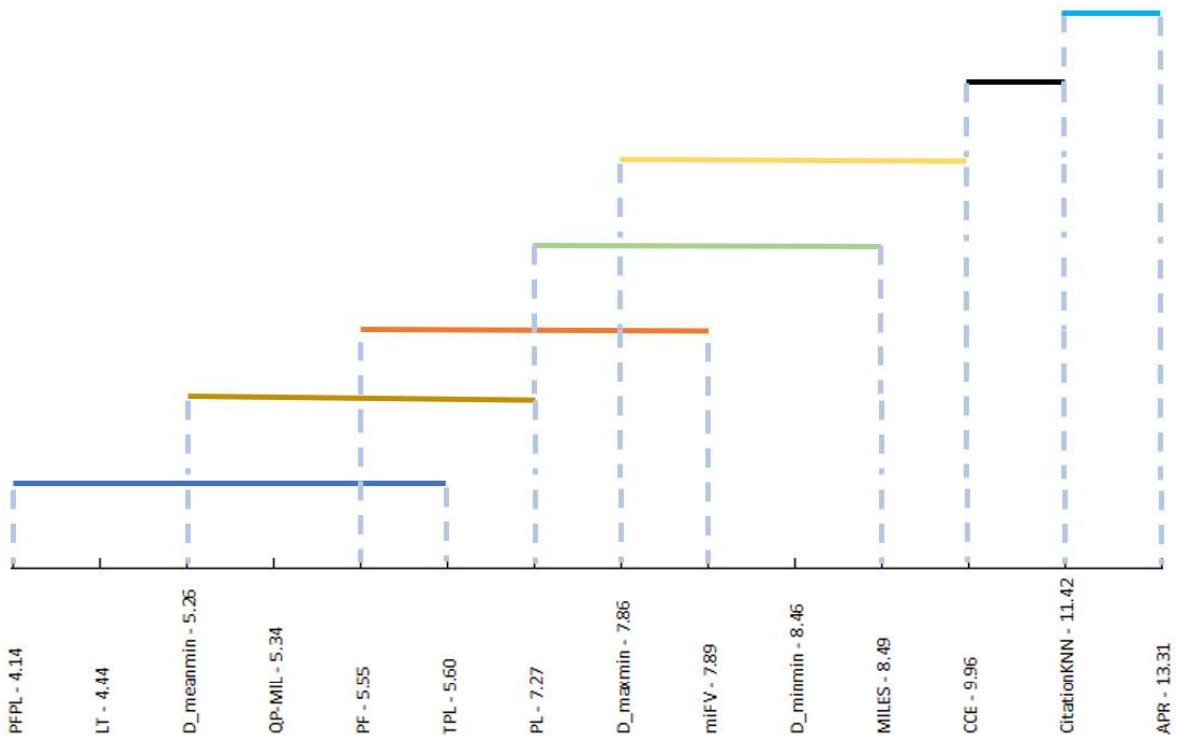


Figure 4.21. Mean AUC performances on benchmark data sets. Numbers indicate the average rank of the approach.

Friedman Test [89] is applied to the results to compare the performances of different approaches. This is a non-parametric and distribution-free test and is generally used to detect the differences between the groups in case the dependent variable is ordinal. Null Hypothesis in this test is that the average ranks of the different treatments are the same. Friedman test concludes that the average ranks between the treatments are significantly different than each other at $\alpha = 5\%$ as p-value $\cong 0$. After the rejection of the null hypothesis, Nemenyi post-hoc test [90] is applied to compare all approaches with each other. This test concludes that two different approaches are significantly different than each other if the average ranks of the two approaches are greater than a certain Critical Difference (CD). CD value is 3.48 for 14 treatments and 71 data sets at $\alpha = 5\%$. Therefore, if the average ranks between two treatments are less than this value, it can be concluded that the two treatments are not significantly better than each other.

Comparisons of all approaches can be found in Figure 4.21. PFPL has the best average rank over all data sets with an average of 4.14 whereas the second best-performing approach LT has the average rank of 4.44. $D_{meanmin}$ is the third best performing approach and has an average rank of 5.26. QP-MIL is the fourth best-performing approach with an average of 5.34 among 14 approaches. The fifth best performing approach is PF with an average rank of 5.55. The average rank difference between the first six approaches is not statistically significant as a result of Nemenyi Test.

However, one of the disadvantages of an average rank based performance measurement is that such methods compare the ranks independent of the accuracy metric, namely AUC in our experiments. For example, TPL's AUC value is 0.999 for Corel Mountains data set. As a result, LT's rank is 5, even LTs AUC difference from the best performing model is 0.001. On contrary, for the Web 6 data set, PFPL is the best performing approach with a 0.877 AUC value which is 0.020 higher than TPL, the second best-performing approach. Therefore, we continue our analysis with the average AUC values based on problem categories that are summarized in Table 4.7. Each row represents the average AUC value of the average of 10 fold cross validation that is repeated 5 times for each problem group. When the average AUC values of the approaches are compared to each other, PFPL also has the highest value over all data sets (0.899). The second best performing approach in terms of average AUC is LT with a value of 0.895. TPL and PF are the third and fourth best performing approaches with an average of 0.891 and 0.887 AUC respectively. QP-MIL and PL are the fifth and sixth best performing approaches with an average of 0.865 and 0.853. In other words, all four methods proposed in Chapter 4 outperform benchmark approaches in terms of average AUC. The results show that the proposed approaches in this thesis have a competitive performance in terms of AUC value in all data sets. However, the rank based performance measurement might be misleading, especially for the data sets in that the AUC difference is small between all approaches.

Table 4.7. Results of the algorithms based on problem categories.

Data set	**APR**	**CCE**	**Cit.**	**Dmax**	**Dmean**	**Dmin**	**MIL.**	**miFV**	**QP**	**PL**	**TPL**	**PF**	**PFPL**	**LT**
Musk	0.789	0.831	0.861	0.938	**0.960**	0.944	0.859	0.830	0.957	0.901	0.922	0.925	0.924	0.926
Mutagen	0.480	0.780	0.757	0.705	0.749	0.555	**0.896**	0.888	0.820	0.831	0.842	0.821	0.850	0.891
Protein	0.509	0.643	0.552	0.561	0.523	**0.876**	0.872	0.873	0.851	0.867	0.807	0.796	0.818	0.855
Animals	0.632	0.769	0.734	0.698	0.800	0.823	0.781	0.799	0.840	**0.855**	0.806	0.767	0.820	0.839
Corel	0.638	0.887	0.848	0.942	**0.962**	0.952	0.881	0.899	0.940	0.920	0.935	0.938	0.943	0.948
UCSB	0.569	0.644	0.706	0.725	0.831	0.791	0.823	0.848	0.888	**0.908**	0.810	0.857	0.825	0.901
Ng	0.500	0.731	0.711	0.837	0.851	0.503	0.654	0.640	0.851	0.814	0.850	0.845	0.858	**0.862**
Web	0.566	0.637	0.579	0.515	0.632	0.623	0.738	0.747	0.585	0.743	0.845	0.820	**0.852**	0.809
Birds	0.596	0.835	0.753	0.739	0.833	0.871	0.959	0.963	**0.965**	0.875	0.952	0.962	0.961	0.936
Avg	0.579	0.785	0.745	0.796	0.844	0.749	0.808	0.811	0.865	0.853	0.891	0.887	**0.899**	0.895

Pairwise comparison of PFPL and LT can be found in Figure 4.22. The average AUC performance over all data sets are close in both approaches. Furthermore, when the problem categories are analyzed, a significant difference between PFPL and LT is not observed. As indicated in Figure 4.23, PFPL outperforms PL in most of the data sets. The AUC difference is larger especially in Web data sets. However, the AUC performance of Web data sets increases when the linear model outputs are integrated to tree-based approach, as in LT.

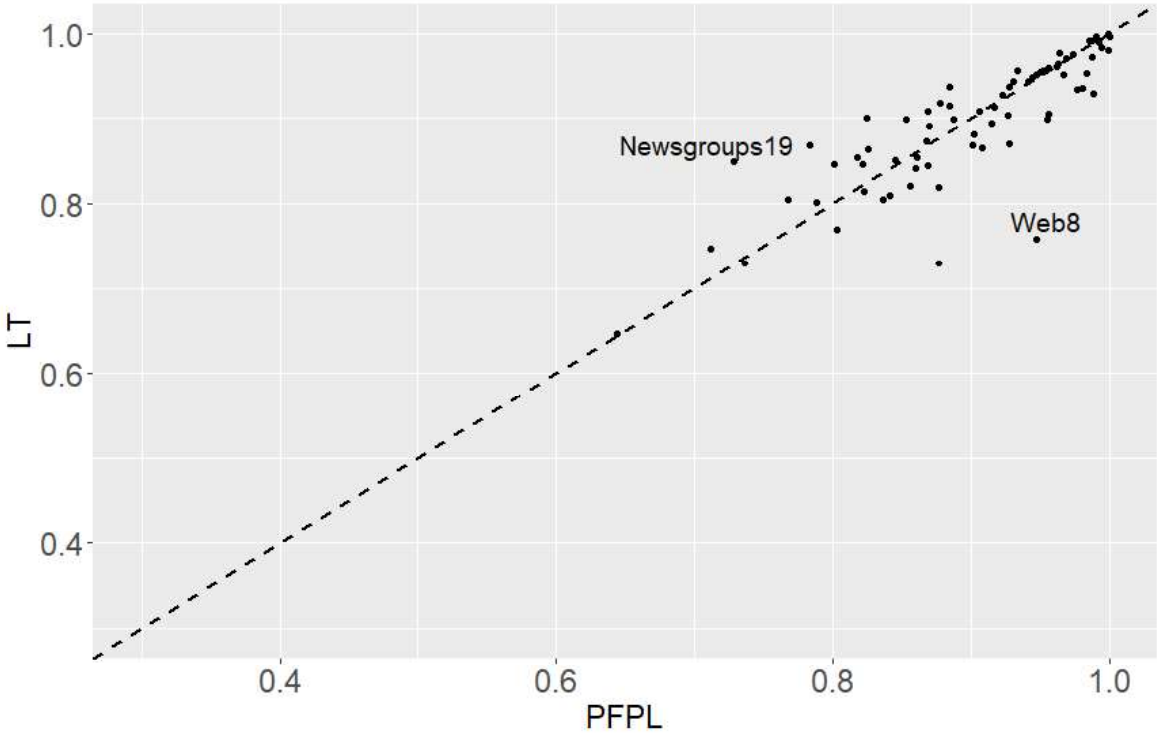


Figure 4.22. PFPL and LT pairwise comparison.

4.8. Parameter Sensitivity

The algorithm's performance is tested under different parameter configurations to discuss the sensitivity of the proposals to algorithm parameters. To show the size of the change and robustness of the PF algorithm, average AUC scores of different parameter settings are reported for the Mutagenesis2 data set. Ten-fold cross-validation is repeated five times for each parameter setting.

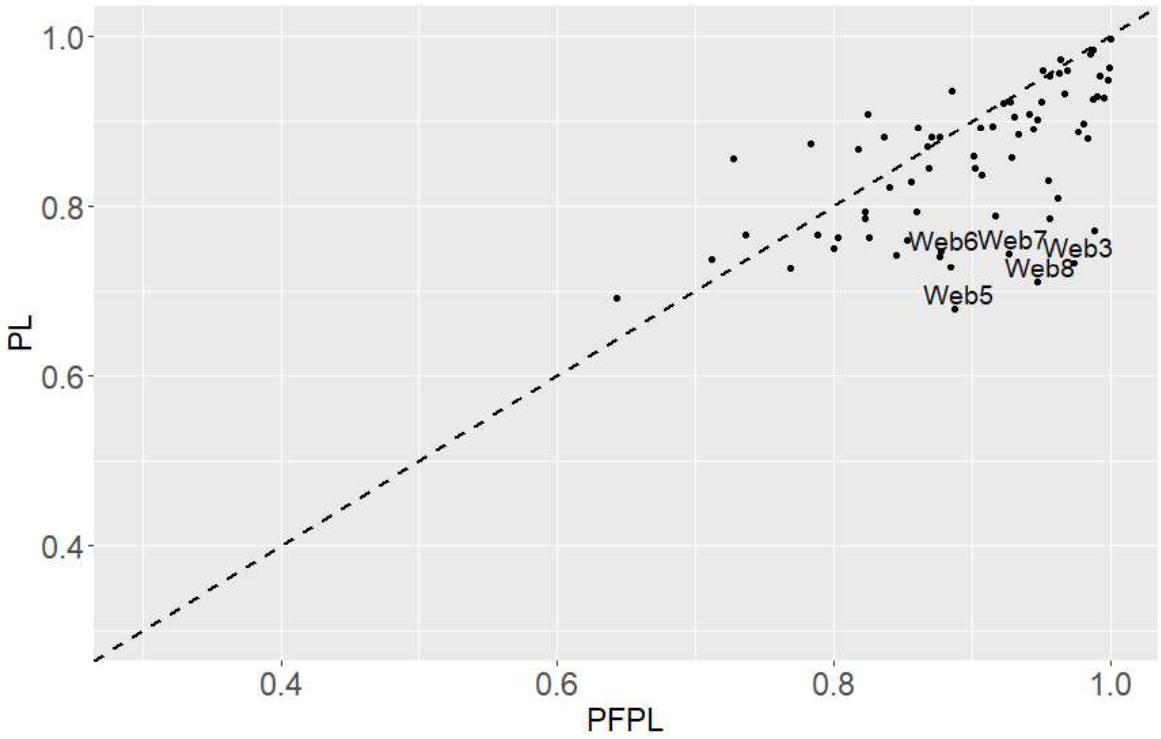


Figure 4.23. PFPL and PL pairwise comparison.

Parameters tested in this section are the number of prototypes D, the number of features F, the number of trees τ, and the maximum depth h. Experiments with the different D and F can be found in Figure 4.24. Effect of four different $D \in \{1, 5, 10, 20\}$ and $F \in \{1, 2, 3\}$ are tested while another parameter configuration is not changed. Average AUC suffers as the number of prototypes increases. Average AUC increases when F is increased. As F denotes different features, this is an expected outcome.

Parameter sensitivity of the number of trees for PF can be found in Figure 4.25. The analysis is run under the set of $h \in \{2, 4, 8\}$ and $\tau \in \{50, 100, 200\}$. The observed algorithm performance increases when h increases. However, algorithm performance is stable with this configuration set of τ.

A similar analysis is completed for PFPL algorithm to test the performances of different configurations. Mutagenesis2 data set is tested for parameter sensitivity of PFPL. Parameters tested for the sensitivity analysis are the number of prototypes D, the number of features F, the number of trees τ, and the maximum depth h. The

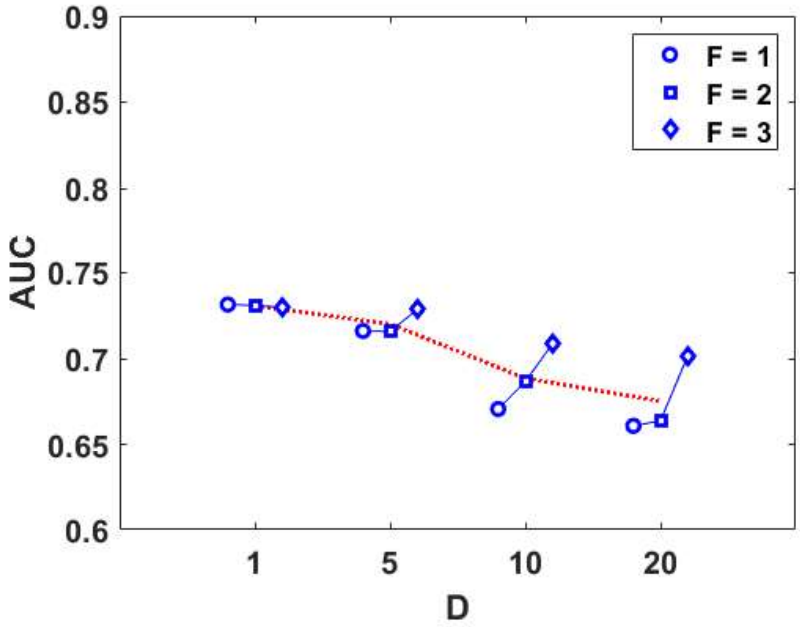


Figure 4.24. PF parameter sensitivity for D vs F.

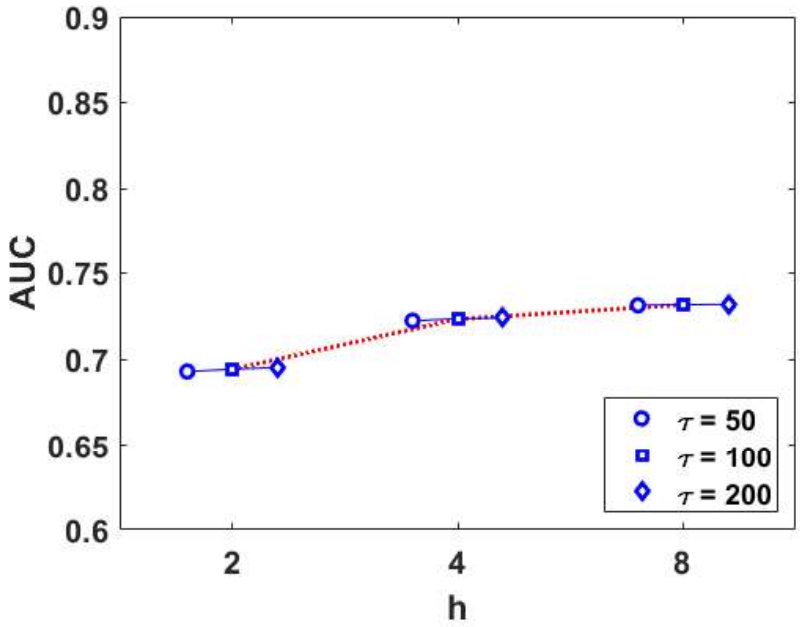


Figure 4.25. PF parameter sensitivity for h vs τ.

same configuration set is used as it is done in PF analysis. The analysis result can be found in Figure 4.26 and Figure 4.27. As it can be observed in Figure 4.26, the algorithm performance decreases when the number of prototypes increases. When

the number of features is decreased, i.e. when a feature is removed, the algorithm performance suffers. Figure 4.27 summarizes that, algorithm performance increases when h increases. Higher τ values also improve the algorithm performance.

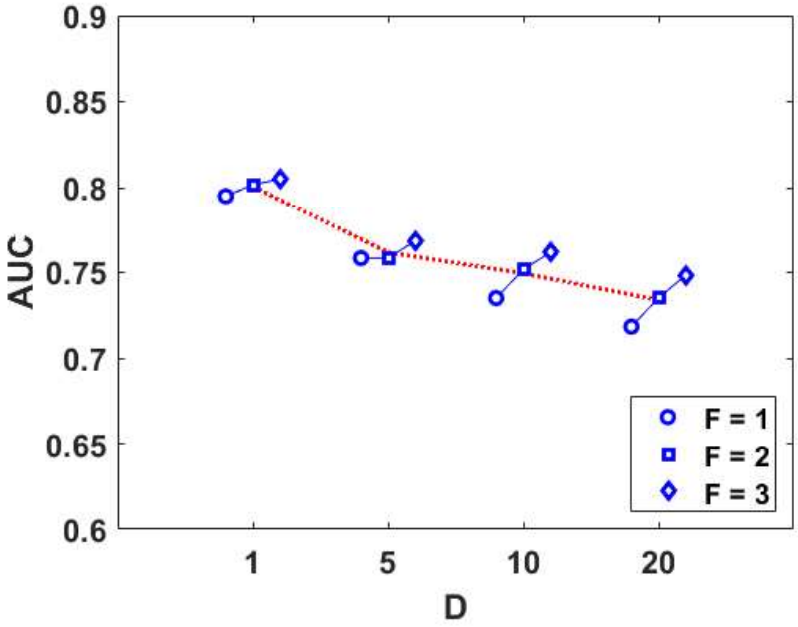


Figure 4.26. PFPL parameter sensitivity for D vs F.

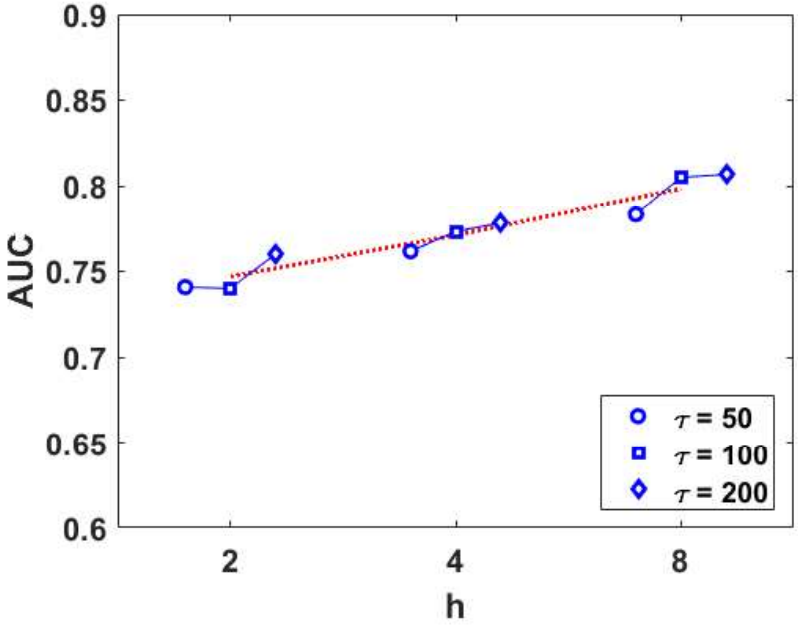


Figure 4.27. PFPL parameter sensitivity for h vs τ.

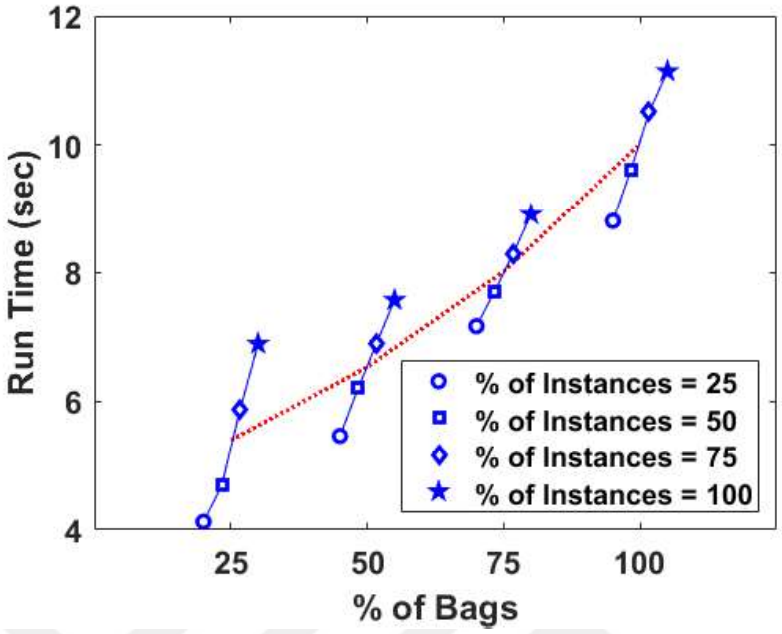


Figure 4.28. Average run time of PL under different problem parameters configurations.

4.9. Time Complexity of the Algorithm

An empirical analysis is conducted to observe the time complexity of the algorithms in real cases. To complete the analysis, two different experiment sets are run on the Mutagenesis2 data set for PF and PFPL. Ten-fold cross-validation is repeated 5 times and the average training run time of each fold is reported. The test run times are omitted due to very fast completion. In the first set of experiments, the effect of problem parameters is tested. The considered problem parameters are the number of bags and the number of instances. 75%, 50%, and 25% of the bags and instances are randomly removed from the data sets respectively. The average run time of the algorithm increases linearly if the number of bags or instances increases as indicated in Figure 4.28 and Figure 4.29.

The second set of experiments is done to analyze the effect of the algorithm parameters on the average run time. The algorithm parameters considered in this set of experiments are h and τ. The same procedure that is used to compare problem param-

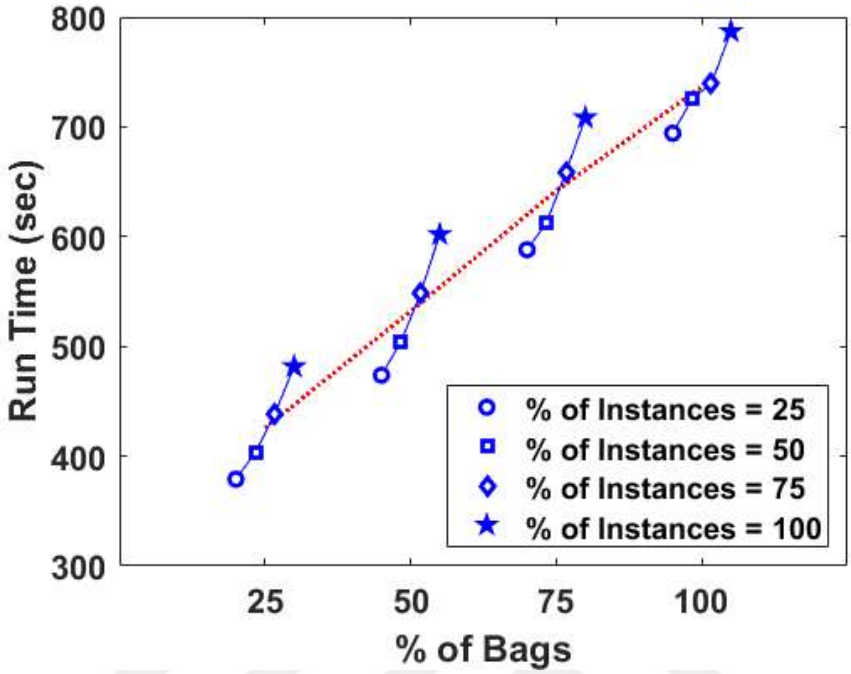


Figure 4.29. Average run time of PFPL under different problem parameters configurations.

eters is repeated to report the average run time under different algorithm parameters. This experiment is run with $h = \{2, 4, 8\}$ $\tau = \{50, 100, 200\}$. As indicated in Figure 4.30 and Figure 4.31, the average run time of the algorithm increases linearly with h and τ.

4.10. Conclusion

This chapter presents a tree-based ensemble framework for MIL problems. The method is tested for benchmark data sets of MIL problems. The results show that PFPL outperforms other well-known methods in MIL literature. One of the main challenges of MIL approaches is the computational complexity and complex parameter tuning processes. PF proposes simplicity, speed, and high prediction power without a complex parameter tuning process. TPL and PFPL extend the work with a prototype learning integration. PFPL proposes the highest prediction performance over all state-of-the-art approaches. Three different distance features based on Euclidean distance

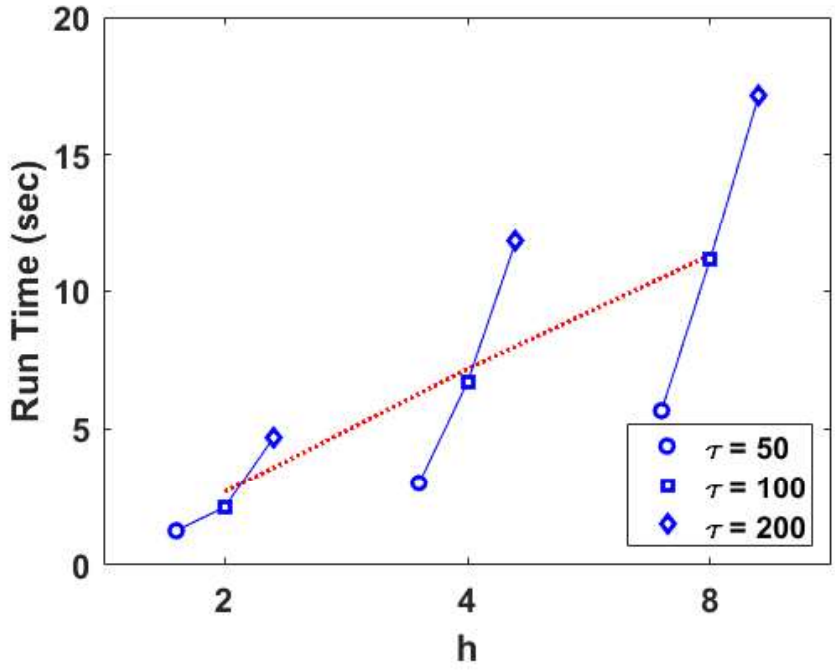


Figure 4.30. Average run time of PF under different h and τ configuration.

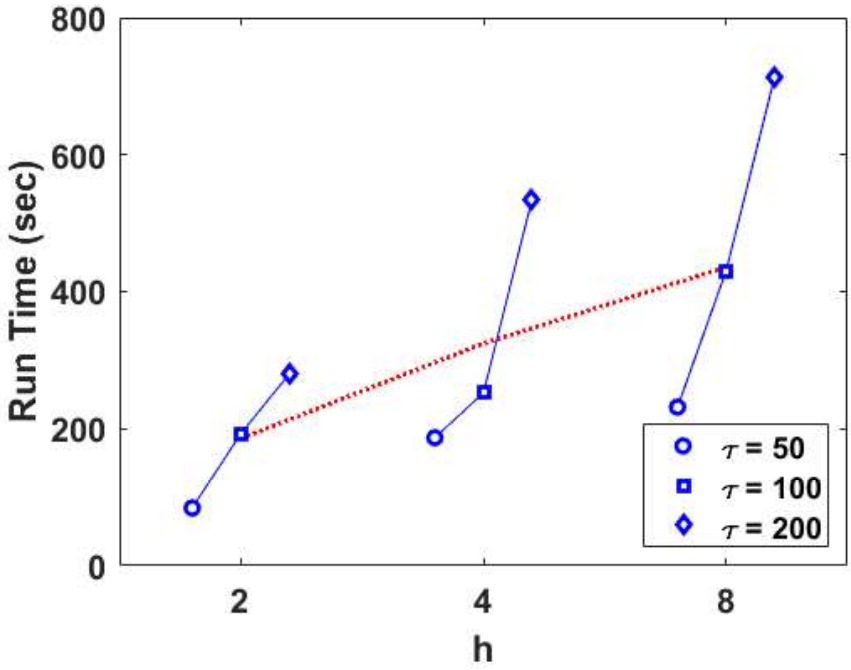


Figure 4.31. Average run time of PFPL under different h and τ configuration.

are used in this paper. As future work, more complex distance features and metrics could be incorporated. Even PFPL, LT, TPL, and PF are applied to benchmark MIL data sets, application in different problems is also a research direction.

5. MULTIPLE INSTANCE REGRESSION

In the previous chapters, we proposed a novel framework in MIL domain. The prediction powers of the proposed approaches are comparable to the well-known approaches of MIL problems where each bag has a discrete label. However, in MIL domain, a real valued responses are also desired to predict. MIR predicts real valued responses in MIL domain. The estimation of crop yields is a great example of this need. The crop yields at the end of a specific time period are real numbers. The production areas, e.g. counties, are represented as bags and patches on the production areas are represented as instances.

This chapter extends the proposed MIL framework to apply the algorithm to MIR domain. The simplicity of the framework enables us to apply the approach in different domains. The loss function and the estimation function of PL can easily be modified for regression problems. Furthermore, updating the split criteria of the tree-based ensemble methods extends the framework to the regression domain. Section 5.1 provides the details of the regression extension of the aforementioned MIL framework.

5.1. Model Formulation

The model formulation for the proposed method is detailed in this section. The other approaches in this thesis concentrate on discrete labels. Therefore the output of a logistic regression is used for the binary classification problems. However, when the response values are real valued numbers the model turns out to be a regression model. The regression model minimizes the mean squared error that can be decomposed for each bag i and solved using a gradient descent algorithm. The decomposed loss is defined as

$$\mathcal{L}_{se}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \tag{5.1}$$

where $\mathcal{L}_{se}(y_i, \hat{y}_i)$ is the squared error. In the equation, y_i is the response and $\hat{y}_i$ is the prediction.

PL algorithm detailed in Chapter 3 is modified for regression problems. The new modification is called Prototype Learning for Multiple Instance Regression (PL-MIR). The cross-entropy loss is replaced by the loss in Equation (5.1) in the objective function of the optimization model. The formulation for the *minimum* features is explained in this section. However, the *maximum* and the *mean* features can be formulated easily.

The mathematical model is expressed as

$$\min_{P,\beta} \sum_{i=1}^{N} \mathcal{L}_{se}(y_i, \hat{y}_i) + \lambda_w \|\beta\|_1 - \lambda_p \sum_{d=1}^{D} \sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{M} \sum_{d=1}^{D} \phi_{id} \tag{5.2}$$

$$\text{st} \qquad \Phi_{id} = \min_{X_{ij} \in B_i} Dist(X_{ij}, P_d) \tag{5.3}$$

$$\hat{y}_i = \beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id} \tag{5.4}$$

where Equation (5.2) is the objective function of the model. The objective function is minimized over prototypes, P, and regression weights, β. The regression weights and the prototypes are regularized. Equation (5.3) defines the distance between the bags and prototypes. The distance between a Bag i to Prototype d is the minimum of the pairwise distances of all instances of Bag i to Prototype d. Equation (5.4) is the estimation of real valued response of Bag i.

The details of the PL-MIR algorithm can be found in Figure 5.1. The algorithm minimizes Equation (5.2) with stochastic gradient algorithm. The distance features are calculated at each step. The distance features are fed to the algorithm to learn update the prototypes and the classifier weights with a learning rate. Mean squared error is calculated with the updated values. If MSE is lower than best error, MSE is assigned as the new best error as indicated in Step 13. There are two stopping conditions of the algorithm. The first is to iterate the algorithm $MaxIter$ times. The second stopping condition is no improvement in three consecutive iterations.

Definitions:

$\lambda_d, \lambda_p, \lambda_w$: Regularization parameters for distances between the prototypes and the bags, the prototypes and the linear classifier weights, α_w : Learning rate of classifier, α_p : Learning rate of prototypes, MaxIter: Number of iterations

foreach *Fold* **do**

  Initialize prototypes;

  BestLoss $\leftarrow M, counter \leftarrow 0$;

  while $iter \leq MaxIter$ **do**

    Loss $\leftarrow \sum_{i=1}^{N} \mathcal{L}_{se}(y_i, \hat{y}_i) + \lambda_w \|\beta\|_1 - \lambda_p \sum_{d=1}^{D} \sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{M} \sum_{d=1}^{D} \phi_{id}$;

    Run Optimizer [83]: Minimize Loss, Update prototypes with α_p, Update regression weights with α_w;

    Calculate features;

    Calculate targets;

    MSE $\leftarrow 1/N \sum_{i=1}^{N} \mathcal{L}_{se}(y_i, \hat{y}_i)$

    if $MSE \leq BestLoss$ **then**

      $BestLoss \leftarrow MSE, counter \leftarrow 0$

    end

    if *Iter % Learning Rate Update = 0* **then**

      $\alpha_p \leftarrow \alpha_p/2, \alpha_w \leftarrow \alpha_w/2$

    end

  end

end

Figure 5.1. PL-MIR algorithm.

The hyperparameters of the PL-MIR algorithm are the regularization parameters of the model and the learning rates. Regularization parameters of distances between the prototypes and the bags, linear regression weights and prototypes are λ_d, λ_w and

λ_p respectively. Similar to PL algorithm the motivation behind the regularization of prototypes is to avoid extreme values for the features of prototypes. The learning rate of the prototypes is α_p and the learning rate of linear regression weights is defined as α_w.

PF-MIR algorithm is the modified version of PF that is detailed in Figure 4.6. The split criteria of PF algorithm is changed to squared error to apply the algorithm to regression domain. The MIR extension of TPL algorithm is called TPL-MIR. The loss function and split criteria of TPL base algorithm (detailed in Figure 4.8) is changed to apply TPL to regression domain. The extension is called TPL-MIR. The prototype learning step of TPL-MIR algorithm can be found in Figure 5.1. Similar to TPL, TPL-MIR tunes the hyperparameters of PL-MIR and constructs a single decision tree with the tuned parameters.

PFPL-MIR algorithm is a tree-based ensemble that constructs several decision trees with the randomly selected hyperparameters of PL-MIR. Split criterion of decision trees are the squared error. The final prediction is made with the median of the predictions of all decision trees. This prediction strategy is applied all tree-based ensemble methods. The details of TPL, PFPL, and LT algorithms are detailed in Chapter 4. All algorithms are modified to apply MIR problems.

5.2. Experimental Results

PL-MIR, PF-MIR, TPL-MIR, PFPL-MIR, and LT-MIR algorithms are tested on generated synthetic data sets and Crop Yield prediction [14] data set. Synthetic data sets are created considering the number of bags in each data set, number of instances in each bag and number of features in each instance. The main motivation of the synthetic data is to validate the approach. We introduced 100, 200, and 400 bags that have 50, 100, and 200 instances. Each instance has 25, 50, and 100 features. To create the data set, a witness instance is created at first. Features of the witness follow a normal distribution with $\mathcal{N}(0, 0.01)$. Gaussian noise is added to the features. After

the witness is created, noisy instances are added to the bags to create MI data.

Definitions:

$tree(\hat{\chi}, depth, D)$, where $\hat{\chi} = \{B_{\hat{i}} : \hat{i} = 1, \ldots, N\}$, is a subset of all bags $\chi = \{B_i : i = 1, \ldots, M\}$. D is the number of prototypes.

F is a feature matrix. $F = \{F_{min}, F_{max}, F_{mean}\}$ is a $N \times (3 * D)$ matrix with $\hat{i}j$th entry is $s_{\hat{i}j}$

if *cannot further split or depth* $= h$ **then**

    Designate this node as a terminal node

    Return

end

Learn D prototypes with randomly selected hyperparameters, extract features F

Obtain p^*, the split point for column p

$\hat{\chi}_{left} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}p} \leq p^*$

$\hat{\chi}_{right} \leftarrow$ bags (rows) $\hat{i}$ with $s_{\hat{i}p} > p^*$

$depth = depth + 1$

$tree(\hat{\chi}_{left}, depth, D)$

$tree(\hat{\chi}_{right}, depth, D)$

Figure 5.2. PFPL-MIR algorithm.

Crop Yield prediction data set consists of wheat and corn yields in Kansas between 2001 and 2004. The bags represent counties in the related states and each instance represents randomly selected MISR pixels (250m × 250m) in the counties. There are some noisy instances that do not have any crops on it (e.g. water or forest). The remote sensing observations are the features of the data set. Red and Infrared reflectances are recorded every 8 days from the selected pixels. However, the first 15 days (30 features) and last 11 days (22 features) are noisy observations as denoted in [72]. Therefore, these days are removed from the data set. Final data set has 20 days of observations, namely 40 features.

The benchmark models tested are Aggregated-MIR, Instance-MIR [13,71], Cluster-MIR [13], Prime-MIR [12], and Pruning-MIR [71]. All benchmark models are implemented in R version 3.6.0. Global pruning that removes $r\%$ of the noisiest instances among all instances was applied for Pruning MIR algorithm. r was set to 5% as it is reported in [71]. Therefore, 5% of the instances removed from the training data set at each iteration until convergence. Mean and median of the instance level predictions are assigned as the bag labels and the parameter that has the lower training error is reported in the Table 5.2. The number of clusters of Cluster MIR is selected with grid search. Finally, in the $Mean$ model, average of the training labels are assigned to the test bags.

5-fold cross-validation is repeated 10 times in the experiments. The bags are randomly assigned to the cross-validation folds. Average RMSE of each fold and each replication is reported as a performance metric. The experiments were done on a Windows Server 2016 operating system. The installed memory of the system is 16.0 GB memory (RAM) and the system has a 2.30 GHz Intel Xeon(R) Processor. The results of the benchmark and the proposed approaches on synthetic data set can be found in Table 5.1. All proposed methods outperform the benchmark. PFPL-MIR and LT-MIR are the best performing approaches. The synthetic data set labels are correlated to the bag to prototype distances and the candidate prototypes are selected among a predetermined prototype set. Therefore, we expect that the prototype learning methods outperforms pure tree based ensemble methods, namely PF.

The results of the crop yield prediction data set can be found in Table 5.2. The results show that the proposed framework outperforms well-known methods in the literature. PFPL-MIR is the best-performing model in both crop and wheat yield data sets.

Table 5.1. Results of the algorithms.

#Bags	#Features	#Instances	Benchmark	PF-MIR	PL-MIR	TPL-MIR	PFPL-MIR	LT-MIR
100	25	50	1.350	1.047	0.974	0.841	0.829	0.972
100	25	100	1.234	0.990	1.021	0.946	0.918	0.865
100	25	200	1.195	0.939	0.915	0.891	0.899	0.846
100	50	50	1.228	0.936	0.976	0.916	0.993	0.931
100	50	100	1.209	0.926	0.991	0.833	0.881	0.924
100	50	200	1.149	0.893	0.985	0.912	0.874	0.859
100	100	50	1.198	0.888	1.017	0.930	0.929	0.834
100	100	100	1.155	0.883	1.029	0.901	0.956	0.918
100	100	200	1.093	0.842	0.975	0.888	0.891	0.832
200	25	50	1.375	1.116	0.977	0.946	0.881	0.906
200	25	100	1.251	0.989	0.982	0.922	0.879	0.811
200	25	200	1.087	0.837	0.889	0.890	0.957	0.938
200	50	50	1.280	0.992	0.958	0.905	0.843	0.912
200	50	100	1.198	0.944	0.956	0.910	0.817	0.951
200	50	200	1.083	0.803	1.010	0.913	0.908	0.934
200	100	50	1.211	0.923	0.101	0.911	0.812	0.946
200	100	100	1.063	0.806	0.997	1.091	0.938	0.904

Table 4.6. Results of the algorithms. (cont.)

#Bags	**#Features**	**#Instances**	**Benchmark**	**PF-MIR**	**PL-MIR**	**TPL-MIR**	**PFPL-MIR**	**LT_MIR**
200	100	200	1.168	0.916	0.974	0.901	0.837	0.794
400	25	50	1.302	1.030	1.009	0.942	0.992	0.923
400	25	100	1.153	0.895	0.963	0.894	0.827	0.811
400	25	200	1.249	0.994	0.973	0.920	1.088	0.933
400	50	50	1.292	1.011	1.102	0.903	0.894	1.041
400	50	100	1.173	0.912	0.984	0.932	0.818	0.908
400	50	200	1.087	0.831	1.014	0.941	0.884	0.861
400	100	50	1.279	1.001	0.980	0.923	0.839	0.994
400	100	100	1.238	0.976	1.024	0.903	0.881	0.852
400	100	200	1.131	0.874	0.975	0.900	0.879	0.872
Average RMSE			1.201	0.933	0.954	0.915	0.894	0.899

Table 5.2. Crop yield prediction results.

Model	**Wheat Yields**	**Corn Yields**
Mean	11.15 ±1.52	39.34 ±1.18
Aggregated MIR	7.13 ±0.31	29.13 ±2.14
Instance MIR	7.35 ±1.11	30.84 ±1.81
Prime MIR	7.33 ±1.23	30.39 ±2.39
Pruning MIR	8.28 ±1.88	31.32 ±3.25
Cluster MIR	6.49 ±0.60	29.02 ±2.25
PF-MIR	6.10 ±0.92	25.70 ±2.57
PL-MIR	6.41 ±1.09	27.64 ±1.95
TPL-MIR	6.17 ±0.98	26.26 ±2.37
PFPL-MIR	5.98 ±1.16	25.12 ±1.84
LT-MIR	6.12 ±1.23	25.91 ±2.21

6. CONCLUSION AND FUTURE RESEARCH

This thesis presents a novel framework for multiple instance learning domain. The proposed learning approaches represent bags considering their distances to the reference objects, namely prototypes. The approaches outperforms state-of-the-art methods in the literature on an extensive benchmark data sets vary from image recognition to molecular activity prediction. The framework's flexibility enables us to combine the tree based ensemble methods and prototype learning approaches. Therefore, the model promises robustness, speed and performance superiority in both classification and regression problems.

PL presents a prototype learning framework for MIL problems that proposes a solution to the two main challenges in MIL literature. The first challenge is that MIL literature lacks interpretable approaches. PL offers the interpretability of the solutions. We applied our approach to a well-known MNIST problem to show the interpretability power of our approach. As a result of this example, we can learn perfectly interpretable prototypes. The second challenge is obtaining robust results on various MIL problems. Certain approaches, including state-of-the-art models that PL is compared to, outperforms in specific problem cases. However, the experiments show that these methods may suffer in other problem types. PL provides accurate and robust results on benchmark MIL data sets when compared to the well-known approaches. We focused on the simplicity and flexibility of the architecture to apply PL to data from all kinds of domains. As a result, PL applies to multi-class problems with simple modifications due to its flexibility. As a future research, the same framework can be extended to the multi-class cases utilizing a softmax function instead of a sigmoid function. Finally, PL has a linear time complexity which results in fast training and test times. However, to improve the run time performance of the model, an efficient batch optimization [100–103] application to PL algorithms is another future research direction.

PF is a tree-based ensemble framework for MIL problems. One of the main challenges of MIL approaches is the computational complexity and complex parameter tuning processes. PF proposes simplicity, speed, and high prediction power without a complex parameter tuning process. TPL, PFPL and LT extend the work with a prototype learning integration. TPL builds a single decision tree with the tuned parameters of PL. The parameters of PL is randomly selected among a configuration set in PFPL. LT utilizes linear model outputs to create splits in decision trees. TPL, PFPL and LT learn prototypes at each node to improve the algorithm performance. All five approach detailed in this thesis outperforms well-known methods in terms of average AUC value. As future work, considering the flexibility of the architecture, one could incorporate more complex classifiers, different distance metrics, and different aggregation procedures to obtain alternative models. The method's modification to different problems is also another research direction. We aim to present the simplicity and flexibility of the approach. Therefore, logistic regression classifiers and Euclidean distance are adopted in this thesis.

In Chapter 5, classification approaches are modified to apply the regression domain. MIR is a less studied area in the multiple instance domain. However, the real valued responses are also needed to be estimated in MIL, e.g. predictions of crop yields. The benchmark models are implemented to compare the performances to our proposals. The proposed approaches outperform other state-of-the-art methods when the average RMSE values are compared. MIR domain has limited literature. Hence, the data sets are not as organized and accessible as the classification domain. Similarly, the implementations of existing methods are not easy to access. Therefore, the improvement of MIR database and implementation of the existing models are interesting future research directions that ease the benchmarking of the new proposals.

REFERENCES

1. Lecun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based Learning Applied to Document Recognition", *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

2. Chen, Y., J. Bi and J. Z. Wang, "MILES: Multiple-Instance Learning via Embedded Instance Selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 12, pp. 1931–1947, 2006.

3. Zhou, Z., Y. Sun and Y. Li, "Multi-instance Learning by Treating Instances as Non-iid Samples", *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1249–1256, Long Beach, USA, 2009.

4. Ilse, M., J. Tomczak and M. Welling, "Attention-based Deep Multiple Instance Learning", Vol. 80 of *Proceedings of Machine Learning Research*, pp. 2127–2136, Stockholm, Sweden, 2018.

5. Maron, O. and T. Lozano-Pérez, "A Framework for Multiple-Instance Learning", *Advances in Neural Information Processing Systems*, Vol. 10, pp. 570–576, 1998.

6. Zhang, Q. and S. Goldman, "EM-DD: An Improved Multiple-instance Learning Technique", *Advances in Neural Information Processing Systems*, Vol. 14, pp. 1073–1080, 2002.

7. Wang, J. and J.-d. Zucker, "Solving the Multiple-instance Problem: A Lazy Learning Approach", *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 1119–1126, Stanford, USA, 2000.

8. Amores, J., "Multiple Instance Classification: Review, Taxonomy and Comparative Study", *Artificial Intelligence*, Vol. 201, pp. 81–105, 2013.

9. Vanwinckelen, G., V. Tragante, D. Fierens and H. Blockeel, "Instance-level Accuracy versus Bag-level Accuracy in Multi-instance Learning", *Data Mining and Knowledge Discovery*, Vol. 30, pp. 313–341, 2015.

10. Carbonneau, M.-A., E. Granger and G. Gagnon, "Thresholding for Accurate Instance Classification in Multiple Instance Learning", *International Conference on Image Processing Theory, Tools and Applications*, pp. 1–6, Oulu, Finland, 2016.

11. Cheplygina, V., D. Tax and M. Loog, "Combining Instance Information to Classify Bags", *Multiple Classifier Systems*, Vol. 7872, pp. 13–24, 2013.

12. Ray, S. and D. Page, "Multiple Instance Regression", *In Proceedings of the 18th International Conference on Machine Learning*, pp. 425–432, Williamstown, USA, 2001.

13. Wagstaff, K. L., T. Lane and A. Roper, "Multiple-Instance Regression with Structured Data", *IEEE International Conference on Data Mining Workshops*, pp. 291–300, Washington, USA, 2008.

14. Wagstaff, K. L. and T. Lane, "Salience Assignment for Multiple-instance Regression", *ICML Workshop on Constrained Optimization and Structured Output Spaces*, Corvallis, USA, 2007.

15. Law, H. C. L., D. J. Sutherland, D. Sejdinovic and S. Flaxman, "Bayesian Approaches to Distribution Regression", *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, Vol. 84, pp. 1167–1176, Lanzarote, Canary Islands, 2018.

16. Szabo, Z., A. Gretton, B. Poczos and B. Sriperumbudur, "Two-stage Sampled Learning Theory on Distributions", *In Proceedings - International Conference on Artificial Intelligence and Statistics*, Vol. 38, pp. 948–957, San Diego, USA, 2015.

17. Sivrikaya, O. E., M. Yuksekgonul and M. G. Baydogan, "Learning Prototypes in

Multiple Instance Learning", *Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 29 (7), pp. 2901–2919, 2021.

18. Alpaydin, E., V. Cheplygina, M. Loog and D. Tax, "Single- vs. Multiple-Instance Classification", *Pattern Recognition*, Vol. 48, pp. 2831–2838, 2015.

19. Dietterich, T., R. H. Lathrop and T. Lozano-Pérez, "Solving the Multiple Instance Problem with Axis-Parallel Rectangles", *Artificial Intelligence*, Vol. 89, pp. 31–71, 1997.

20. Andrews, S., T. Hofmann and I. Tsochantaridis, "Multiple Instance Learning with Generalized Support Vector Machines", *Proceedings of the National Conference on Artificial Intelligence*, pp. 561–568, Edmonton, Canada, 2002.

21. Carbonneau, M.-A., E. Granger, A. Raymond and G. Gagnon, "Robust Multiple-instance Learning Ensembles Using Random Subspace Instance Selection", *Pattern Recognition*, Vol. 58, pp. 83–99, 2016.

22. Foulds, J. and E. Frank, "A Review of Multi-Instance Learning Assumptions", *The Knowledge Engineering Review*, Vol. 25, pp. 1–25, 2010.

23. Doran, G. and S. Ray, "Learning Instance Concepts from Multiple-instance Data with Bags as Distributions", *Proceedings of the National Conference on Artificial Intelligence*, Vol. 3, pp. 1802–1808, Québec City, Canada, 2014.

24. Rahmani, R. and S. Goldman, "MISSL: Multiple-instance Semi-supervised Learning", *Machine Learning, Proceedings of the Twenty-Third International Conference*, Vol. 148, pp. 705–712, Pittsburgh, USA, 2006.

25. Li, Z., G.-H. Geng, J. Feng, J.-y. Peng, C. Wen and J.-l. Liang, "Multiple Instance Learning Based on Positive Instance Selection and Bag Structure Construction", *Pattern Recognition Letters*, Vol. 40, pp. 19–26, 2014.

26. Carbonneau, M.-A., V. Cheplygina, E. Granger and G. Gagnon, "Multiple Instance Learning: A Survey of Problem Characteristics and Applications", *Pattern Recognition*, Vol. 77, pp. 329–353, 2018.

27. LI, W.-X. and N. Vasconcelos, "Multiple Instance Learning for Soft Bags via Top Instances", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4277–4285, Boston, USA, 2015.

28. Weidmann, N., E. Frank and B. Pfahringer, "A Two-level Learning Method for Generalized Multi-instance Problems", *Lecture Notes in Artifical Intelligence*, Vol. 2837, pp. 468–479, 2003.

29. Li, Y., D. M. Tax, R. P. Duin and M. Loog, "Multiple-instance Learning as a Classifier Combining Problem", *Pattern Recognition*, Vol. 46, No. 3, pp. 865–874, 2013.

30. Zhang, C., J. C. Platt and P. A. Viola, "Multiple Instance Boosting for Object Detection", *Advances in Neural Information Processing Systems*, Vol. 18, pp. 1417–1424, 2006.

31. Liu, G., J. Wu and Z.-H. Zhou, "Key Instance Detection in Multi-Instance Learning", *Journal of Machine Learning Research*, Vol. 25, pp. 253–268, 2012.

32. Xu, X., *Statistical Learning in Multiple Instance Problems*, Master's Thesis, University of Waikato, 2003.

33. Xu, X. and E. Frank, "Logistic Regression and Boosting for Labeled Bags of Instances", *Proceedings of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 272–281, Sydney, Australia, 2004.

34. Raykar, V., B. Krishnapuram, J. Bi, M. Dundar and R. Rao, "Bayesian Multiple Instance Learning: Automatic Feature Selection and Inductive Transfer", *Proceedings of the 25th International Conference on Machine Learning*, pp. 808–815,

Helsinki, Finland, 2008.

35. Fu, Z., A. Robles-Kelly and J. Zhou, "MILIS: Multiple Instance Learning with Instance Selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, pp. 958–977, 2011.

36. Ray, S. and M. Craven, "Supervised versus Multiple Instance Learning: An Empirical Comparison", *Proceedings of 22nd International Conference on Machine Learning*, pp. 697–704, Bonn, Germany, 2005.

37. Tax, D., M. Loog, R. Duin, V. Cheplygina and W.-J. Lee, "Bag Dissimilarities for Multiple Instance Learning", *International Workshop on Similarity-Based Pattern Recognition*, Vol. 7005, pp. 222–234, Venice, Italy, 2011.

38. Gartner, T., P. Flach, A. Kowalczyk and A. Smola, "Multi-Instance Kernels", *Proceedings of the 19th International Conference on Machine Learning*, pp. 179–186, Sydney, Australia, 2002.

39. Tax, D., E. Hendriks, M. Valstar and M. Pantic, "The Detection of Concept Frames Using Clustering Multi-instance Learning", *International Conference on Pattern Recognition*, pp. 2917–2920, Istanbul, Turkey, 2010.

40. Zhou, Z., K. Jiang and M. Li, "Multi-instance Learning based Web Mining", *Applied Intelligence*, Vol. 22(2), pp. 135–147, 2005.

41. Zhou, Z. H. and J. M. Xu, "On the Relation between Multi-instance Learning and Semi-supervised Learning", *Proceedings of the 24th International Conference on Machine Learning*, pp. 1167–1174, Corvallis, USA, 2007.

42. Pekalska, E. and R. P. W. Duin, *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, World Scientific, 2005.

43. Snell, J., K. Swersky and R. Zemel, "Prototypical Networks for Few-shot Learn-

ing", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 4080–4090, 2017.

44. Cheplygina, V., D. M. Tax and M. Loog, "Multiple Instance Learning with Bag Dissimilarities", *Pattern Recognition*, Vol. 48, No. 1, pp. 264–275, 2015.

45. Rubner, Y., C. Tomasi and L. Guibas, "The Earth Mover's Distance as a Metric for Image Retrieval", *International Journal of Computer Vision*, Vol. 40, pp. 99–121, 2000.

46. Cheplygina, V., D. M. J. Tax and M. Loog, "Dissimilarity-Based Ensembles for Multiple Instance Learning", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 27, No. 6, pp. 1379–1391, 2016.

47. Briggs, F., B. Lakshminarayanan, L. Neal, X. Z. Fern, R. Raich, S. J. K. Hadley, A. S. Hadley and M. G. Betts, "Acoustic Classification of Multiple Simultaneous Bird Species: A Multi-instance Multilabel Approach", *The Journal of the Acoustical Society of America*, Vol. 131, pp. 4640–4650, 2012.

48. Gehler, P. and O. Chapelle, "Deterministic Annealing for Multiple-Instance Learning", *Proceedings of the Eleventh International Conference Artificial Intelligence and Statistics*, pp. 123–130, San Juan, Puerto Rico, 2007.

49. Wang, X., Y. Yan, P. Tang, X. Bai and W. Liu, "Revisiting Multiple Instance Neural Networks", *Pattern Recognition*, Vol. 74, pp. 15–24, 2018.

50. Phan, S., D.-D. Le and S. Satoh, "Multimedia Event Detection Using Event-Driven Multiple Instance Learning", *Proceedings of the 23rd ACM International conference on Multimedia*, pp. 1255–1258, Brisbane, Australia, 2015.

51. Cinbis, R., J. Verbeek and C. Schmid, "Weakly Supervised Object Localization with Multi-Fold Multiple Instance Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, pp. 189–203, 2015.

52. Wu, J., Y. Yu, C. Huang and K. Yu, "Deep Multiple Instance Learning for Image Classification and Auto-annotation", *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3460–3469, Boston, USA, 2015.

53. Tang, P., X. Wang, S. Bai, W. Shen, X. Bai, W. Liu and A. L. Yuille, "PCL: Proposal Cluster Learning for Weakly Supervised Object Detection", ArXiv: 1807.03342, 2018.

54. Wan, F., C. Liu, W. Ke, X. Ji, J. Jiao and Q. Ye, "C-MIL: Continuation Multiple Instance Learning for Weakly Supervised Object Detection", ArXiv: 1904.05647, 2019.

55. Dennis, D. K., C. Pabbaraju, H. V. Simhadri and P. Jain, "Multiple Instance Learning for Efficient Sequential Data Classification on Resource-constrained Devices", *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 10976–10987, Montréal, Canada, 2018.

56. Angelidis, S. and M. Lapata, "Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis", ArXiv:1711.09645, 2017.

57. McFee, B., J. Salamon and J. P. Bello, "Adaptive Pooling Operators for Weakly Labeled Sound Event Detection", ArXiv: 1804.10070, 2018.

58. Bergeron, C., G. Moore, J. Zaretzki, C. M. Breneman and K. P. Bennett, "Fast Bundle Algorithm for Multiple-Instance Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 6, pp. 1068–1079, 2012.

59. Mangasarian, O. L. and E. W. Wild, "Multiple Instance Classification via Successive Linear Programming", *Journal of Optimization Theory and Applications*, Vol. 137, No. 3, pp. 555–568, 2008.

60. Breiman, L., "Random Forests", *Machine Learning*, Vol. 45, pp. 5–32, 2001.

61. Zucker, J.-d. and Y. Chevaleyre, "Solving Multiple-instance and Multiple-part Learning Problems with Decision Trees and Decision Rules. Application to the Mutagenesis Problem", *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pp. 204–214, Ottawa, Canada, 2001.

62. Frank, E. and B. Pfahringer, "Propositionalisation of Multi-instance Data Using Random Forests", *Australasian Conference on Artificial Intelligence*, Vol. 8272, pp. 362–373, Dunedin, New Zealand, 2013.

63. Kucukasci, E. and M. Baydogan, "Bag Encoding Strategies in Multiple Instance Learning Problems", *Information Sciences*, Vol. 467, pp. 559–578, 2018.

64. Straehle, C., M. Kandemir, U. Köthe and F. Hamprecht, "Multiple Instance Learning with Response-Optimized Random Forests", *International Conference on Pattern Recognition*, pp. 3768–3773, Stockholm, Sweden, 2014.

65. Blockeel, H. and A. Srinivasan, "Multi-instance Tree Learning", *22nd International Conference on Machine Learning*, pp. 57–64, Bonn, Germany, 2005.

66. Ni, B., Z. Song and S. Yan, "Web Image and Video Mining Towards Universal and Robust Age Estimator", *IEEE Transactions on Multimedia*, Vol. 13, No. 6, pp. 1217–1229, 2011.

67. Dooly, D. R., Q. Zhang, S. A. Goldman and R. A. Amar, "Multiple-Instance Learning of Real-Valued Data", *Journal of Machine Learning Research*, Vol. 3, pp. 3–10, 2001.

68. Kwok, J. T. and P. M. Cheung, "Marginalized Multi-instance Kernels", *20th International Joint Conference on Artificial Intelligence*, pp. 901–906, Hyderabad, India, 2007.

69. Ray, S., *Learning from Data with Complex Interactions and Ambiguous Labels*,

Ph.D. Thesis, University of Wisconsin, 2015.

70. Pappas, N. and A. Popescu-Belis, "Explaining the Stars: Weighted Multiple-Instance Learning for Aspect-Based Sentiment Analysis", *Conference on Empirical Methods in Natural Language Processing*, pp. 455–466, Doha, Qatar, 2014.

71. Wang, Z., V. Radosavljevic, B. Han and Z. Obradovic, "Aerosol Optical Depth Prediction from Satellite Observations by Multiple Instance Regression", *SIAM International Conference on Data Mining*, pp. 165–176, Atlanta, USA, 2008.

72. Wang, Z., L. Lan, and S. Vucetic, "Mixture Model for Multiple Instance Regression and Applications in Remote Sensing", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 50, pp. 2226—-2237, 2012.

73. Muandet, K., K. Fukumizu, B. Sriperumbudur and B. Schölkopf, *Kernel Mean Embedding of Distributions: A Review and Beyond*, Now Foundations and Trends, 2017.

74. Srinivasan, A., S. Muggleton and R. King, "Comparing the Use of Background Knowledge by Inductive Logic Programming Systems", *5th International Workshop of Inductive Logic Programming*, pp. 199–230, Tokyo, Japan, 1995.

75. Tao, Q., S. Scott, N. Vinodchandran and T. T. Osugi, "Svm-based Generalized Multiple-Instance Learning via Approximate Box Counting", *Twenty-first International Conference on Machine Learning*, p. 101, Banff, Canada, 2004.

76. Andrews, S., T. Hofmann and I. Tsochantaridis, "Support Vector Machines for Multiple-Instance Learning", *Advances in Neural Information Processing Systems*, Vol. 15, pp. 561–568, 2003.

77. Kandemir, M., C. Zhang and F. A. Hambrecht, "Empowering Multiple Instance Hispathology Cancer Diagnosis by Cell Graphs", *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 228–235,

Boston, USA, 2014.

78. J. F. Ruiz-Munoz, M. O.-A. and G. Castellanos-Dominguez, "Multiple Instance Learning-based Birdsong Classification Using Unsupervised Recording Segmentation", *24th International Conference on Artificial Intelligence*, pp. 2632–2638, Buenos Aires, Argentina, 2015.

79. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

80. Ba, L. J., J. R. Kiros and G. E. Hinton, "Layer Normalization", ArXiv: 1607.06450, 2016.

81. Ioffe, S. and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *32nd International Conference on International Conference on Machine Learning*, Vol. 37, pp. 448–456, Lille, France, 2015.

82. LeCun, Y. and C. Cortes, "MNIST Handwritten Digit Database", 2010, `http://yann.lecun.com/exdb/mnist/`, accessed on August 31, 2022.

83. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *3rd International Conference on Learning Representations*, San Diego, USA, 2015.

84. Kucukasci, E. S. and M. G. Baydogan, "Multiple Instance Learning Repository", 2018, `http://www.multipleinstancelearning.com/`, accessed on August 21, 2022.

85. Zhou, Z.-H. and M.-L. Zhang, "Solving Multi-instance Problems with Classifier Ensemble based on Constructive Clustering", *Knowledge and Information Systems*, Vol. 11, No. 2, pp. 155–170, 2007.

86. Wei, X.-S., J. Wu and Z.-H. Zhou, "Scalable Algorithms for Multi-instance Learning", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28,

No. 4, pp. 975–987, 2017.

87. Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic Differentiation in PyTorch", *31st Conference on Neural Information Processing Systems*, Long Beach, USA, 2017.

88. The GPyOpt, A., "GPyOpt: A Bayesian Optimization Framework In Python", 2016, `http://github.com/SheffieldML/GPyOpt`, accessed on August 31, 2022.

89. Friedman, M., "A Comparison of Alternative Tests of Significance for the Problem of m Rankings", *The Annals of Mathematical Statistics*, Vol. 11, No. 1, pp. 86–92, 1940.

90. Nemenyi, P., *Distribution-free Multiple Comparisons*, Princeton University, 1963.

91. Jolliffe, I., *Principal Component Analysis*, Springer Verlag, 1986.

92. Wold, S., K. Esbensen and P. Geladi, "Principal Component Analysis", *Chemometrics and Intelligent Laboratory Systems*, Vol. 2, No. 1, pp. 37–52, 1987.

93. Moosmann, F., B. Triggs and F. Jurie, "Fast Discriminative Visual Codebooks using Randomized Clustering Forests", *Advances in Neural Information Processing Systems*, Vol. 19, pp. 985–992, 2007.

94. Hastie, T., R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer, 2001.

95. Caruana, R., N. Karampatziakis and A. Yessenalina, "An Empirical Evaluation of Supervised Learning in High Dimensions", *25th International Conference on Machine Learning*, pp. 96–103, Helsinki, Finland, 2008.

96. Leistner, C., A. Saffari and H. Bischof, "MIForests: Multiple-Instance Learning with Randomized Trees", *11th European Conference on Computer Vision*, pp.

29–42, Crete, Greece, 2010.

97. Quinlan, J. R., "Induction of Decision Trees", *Machine Learning*, Vol. 1, pp. 81–106, 1986.

98. Frank, E., Y. Wang, S. Inglis, G. Holmes and I. H. Witten, "Using Model Trees for Classification", *Machine Learning*, Vol. 32, pp. 63–76, 1998.

99. Mann, H. B. and D. R. Whitney, "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other", *Annals of Mathematical Statistics*, Vol. 18, No. 1, pp. 50–60, 1947.

100. Li, M., T. Zhang, Y. Chen and A. J. Samola, "Efficient Mini-batch Training for Stochastic Optimization", *20th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 661–670, New York, USA, 2014.

101. Masters, D. and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks", ArXiv: 1804.07612, 2018.

102. Keskar, N. S., D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima", ArXiv: 1609.04836, 2016.

103. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

APPENDIX A: REGULARIZATION

Experiments are repeated for Musk2 and CorelAntique data sets without prototype regularization to compare the algorithm performance. The experiment method and all settings are kept. The only difference is the removal of the prototype regularization from the objective function and the parameters. As a result of experiments, the average AUC value decreased from 0.892 to 0.828 for Musk2 and 0.894 to 0.591 for CorelAntique data sets. Therefore, we can conclude that regularization improves performance of the algorithm when compared to the results without prototype regularization in terms of average AUC value. Prototypes that the model is run with regularization can be found in Figure A.1 and without regularization can be found in Figure A.2. Since the parameters were tuned again, the number of prototypes is different between the PL and the PL without prototype regularization. When we analyze the output prototypes, we can conclude that the regularization term prevents fluctuations and extreme values. The features of the prototypes change between $+2.5$ and -2.5 with regularization whereas the features change between $+5$ and -5 without regularization.

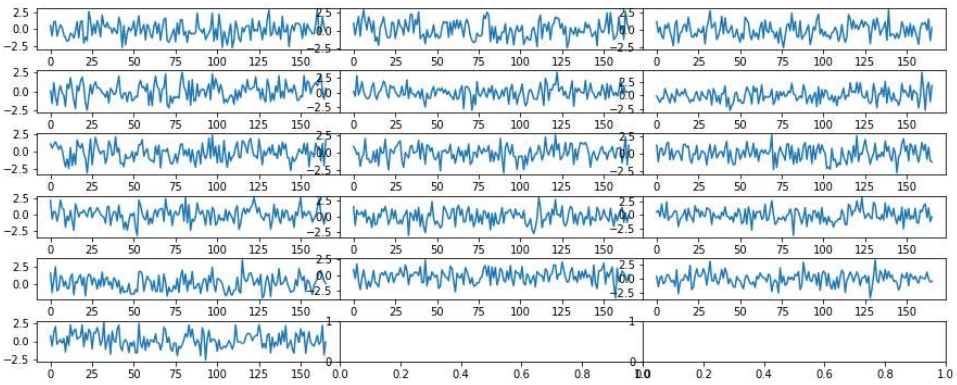


Figure A.1. Musk2 prototypes with regularization.

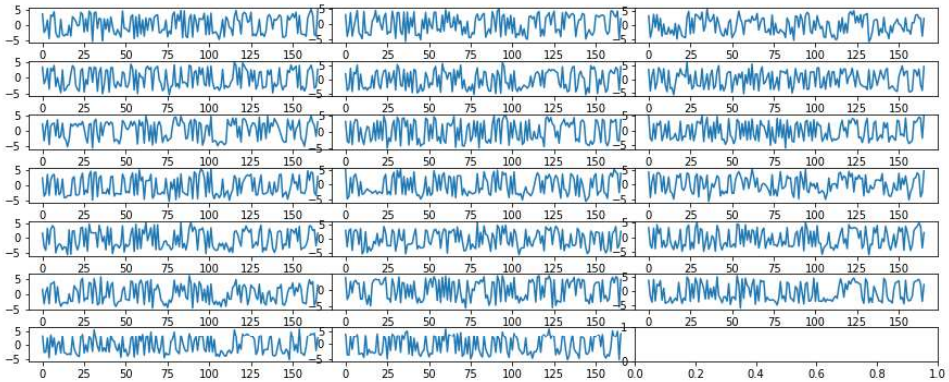


Figure A.2. Musk2 prototypes without regularization.

APPENDIX B: ABOUT THE FIGURES USED IN THE DOCUMENT

For [1], the IEEE does not require individuals working on a thesis to obtain a formal reuse license. The details can be found in the following link: `https://www.ieee.org/content/dam/ieee-org/ieee/web/org/pubs/permissions_faq.pdf`.

The figures that are created in the scope of this thesis and whose copyrights were transferred to the publishing house are used in the thesis in accordance with the "publishing policy valid for the reuse of the text and graphics produced by the author" on the website of the publisher.