Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2901 – 2919
© TÜBİTAK
doi:10.3906/elk-2101-103

Learning prototypes for multiple instance learning

Özgür Emre SİVRİKAYA[1,*], **Mert YÜKSEKGÖNÜL**[1,2], **Mustafa Gökçe BAYDOĞAN**[1]

[1]Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey

[2]Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey

Received: 22.01.2021 • **Accepted/Published Online:** 02.07.2021 • **Final Version:** 30.11.2021

Abstract: Multiple instance learning (MIL) is a weakly supervised learning method that works on the labeled bag of instances data. A prototypical network is a popular embedding approach in MIL. They overcome the common problems that other MIL approaches may have to deal with including dimensionality, loss of instance-level information, and complexity. They demonstrate competitive performance in classification. This work proposes a simple model that provides a permutation invariant 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.

Key words: Multiple instance learning, prototypical networks, interpretability, stochastic gradient descent, dissimilarity, embedding, pattern recognition

1. Introduction

Classification problems can be divided into two concerning the labeling characteristics of the data, single instance (SI) and multiple instance (MI) problems. In single instance learning (SIL) problems, each instance is individually labeled. A common example of SIL problems is detecting spam e-mails. In this setting, each e-mail is an instance represented by a feature vector and is labeled as spam or not. However, multiple instance learning (MIL) concentrates on bags of instances, not individually labeled instance data. Detecting whether an object is on an image or not can be given as an example of MIL problems. There may be several objects in an image. However, if one is focusing on finding a certain object, e.g., an elephant, other objects in the image might be misleading. Therefore, dividing the image into several patches and solving the problem in MIL domain is a good approach. Two different labeling characteristics of classification problems can be seen in Table 1.

Mainly there are two alternative approaches in MIL. The first one is instance and bag level approaches, and the second one is embedding approaches. The instance-level approaches try to reward a probability per each instance that exists in a bag, then apply some pooling function to probabilities to obtain the final bag probability. In the second type of approach, an arbitrary function, often a neural network, is used to come up with an embedding for each instance, then again some pooling function is applied to aggregate information from each embedding which is fed to a classifier. Embedding approaches overcome common problems of instance-level approaches that are summarized in Section 2. Prototypical networks are one of the most popular

*Correspondence: ozgur.sivrikaya@boun.edu.tr

Table 1. Labeling in MIL and SIL problems.

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

Bag	Instance	Label
B_1	X_1 X_2	1
B_2	X_3 X_4 X_5	0

embedding approaches in MIL problems. These networks construct a new feature space that uses the most granular information without dealing with high dimensionality. Then, bag-to-prototype dissimilarities are used to perform classification. Therefore, the selection or learning of prototypes plays a critical role in the algorithm's prediction power.

Another common problem of MIL approaches summarized in this research is the incapability of being interpreted. The main reason behind this is the decrease in the algorithm's prediction accuracy in the interpretable models in general [1]. Even interpretation in complex deep learning methods has been drawing attention [2], MIL literature still lacks interpretable approaches [3]. Interpretation can be defined as the translation of the algorithm behavior into an understandable domain for a human. Therefore, interpretation is critical for a robust algorithm, further exploration, and analysis [4].

One of the main challenges in MIL solution approaches is providing results without sacrificing interpretability [3]. Even interpretation is critical in terms of evaluating results and further development and analysis [4], only a few approaches in MIL literature are capable of generating interpretable results [1, 3, 5]. One other challenge is developing a robust approach that works in several MIL problems. Certain methods summarized in Section 2 outperforms in specific problem cases. However, due to various problem sets in this domain, these methods suffer in certain cases when they are applied to all benchmark problems. The main objective of this study is to provide a basic, robust, and interpretable approach to MIL problems.

In this work, we propose an interpretable embedding approach with some modifications. The idea is to find some representative prototypes in the feature space so that bags are linearly separable when they are represented as their distances to the prototypes. Distance features between bags and prototypes are detailed in subsection 3.1. Details of the study will be given in the following chapters under the following structure. Section 2 gives an overview of the previous studies of the field. Section 3 explains the prototype learning algorithm (PL) and the solution method of the problem. Updates for the initial learning algorithm also in the scope of Section 3. Finally, the results of the solution approach on various data sets and our conclusions can be found in Sections 5 and 6.

2. Literature review

MIL problems have different categorizations based on the instance or bag characteristics [6]. These categories are determined with the instance characteristics. In the first category, all instances carry the bag's information, e.g., a bag of pictures of similar animal images. Therefore, instances can be labeled with the bag's label [5, 7]. In the second category, one or more instances determine the bag's label, i.e. MIL problems with standard assumption and its variations [8–10]. Standard MIL approaches assume that if a bag has at least one positive instance, then the bag's label is positive [11]. 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 a bag's information and each instance contributes to the bag's label. The aforementioned approaches 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.

Another approach, called the embedding approach, overcomes these disadvantages. In these approaches, an arbitrary function, often a neural network, is used to come up with an embedding for each instance, then again some pooling function is applied to aggregate information from each embedding which is fed to a classifier. Dissimilarity is one of the most popular embedding approaches [12–14]. It stands for the representation of an object by describing it relative to a set of reference objects, called prototypes. 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 the MIL domain. In such methods, certain objects can be used as prototypes including bags [13], instances [15] or ensembles [16, 17]. These methods followed by convolutional neural network based prototype learning methods [1, 18, 19] after their great success. In addition to prototype learning methods, the application of neural network based approaches to the MIL problems has been drawing attention from several different domains in recent years [3, 20]. Well known problems have also been reformulated for this particular purpose, such as common computer vision tasks like image classification [21], weakly supervised object detection [22, 23], sequence predictions [24], sentiment analysis [25] and sound event detection [26].

3. Joint learning of prototypes and classification boundary in multiple instance learning

3.1. Definitions

Let X_{ij} be a L-dimensional feature vector of instance j of bag i. Feature r of instance j is referred to as $X_{ij,r}$. A bag i is a group of instances which is defined as $\chi = \{B_i : i = 1, \ldots, M\}$. Each bag is also defined with its label y_i and there are K_i instances in bag i. Each instance has L number of features. L is fixed for all instances in bag i but changes between bags. The number of instances is not fixed for each bag. In MIL, we are searching for a classifier to identify the bag's label. All symbols used and their descriptions can be found in Table 2.

Table 2. Table of notations.

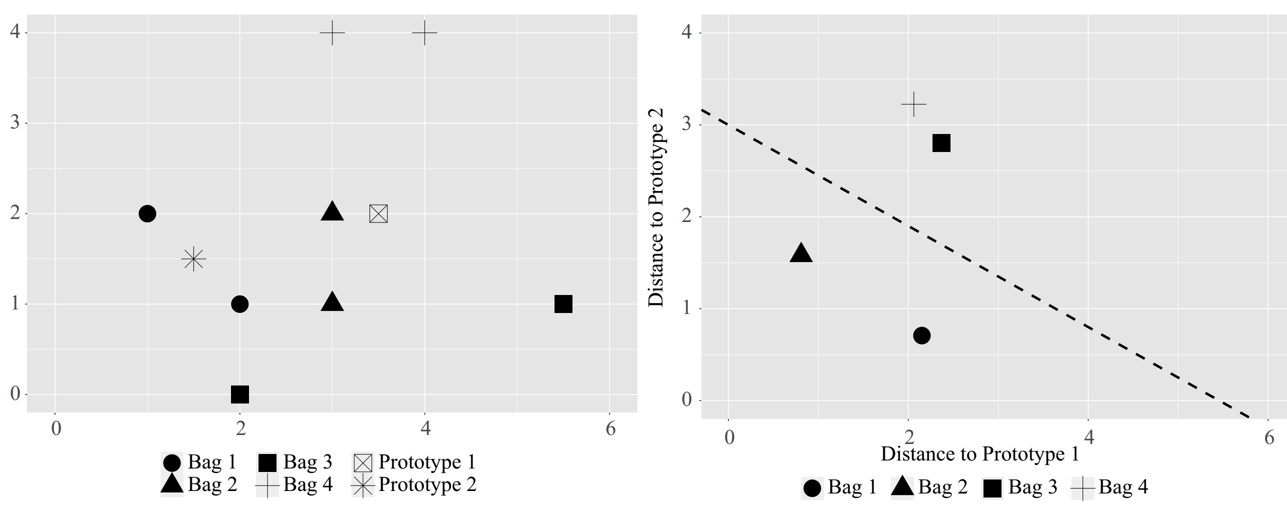
Notation	Description	Notation	Description
B_i	Bag i	M	Number of bags
y_i	Label of bag i	$\hat{Y}_i$	Estimated label of bag i
K_i	Number of instances in bag i	X_{ij}	Instance j of bag i
$X_{ij,r}$	Feature r of instance j	L	Number of features
P_d	Prototype d	D	Number of prototypes
$\sigma(.)$	Sigmoid function	$Dist(.,.)$	Distance
Φ_{id}	Min, max, or average distance between bag i and prototype d	$\mathcal{L}_{ce}(.,.)$	Cross-entropy loss
$\lVert.\rVert$	Norm	μ	Mean
σ	Standard deviation	β	Linear classifier weights

A simple example of MIL representation can be found in Table 3. In this example, there are two bags B_1 and B_2, and each instance has two features. The number of instances that each bag has may vary in MIL problems. In this study, 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 3. Representation of a simple MIL example.

R^2			
Bag	Label	Feature 1	Feature 2
B_1	1	$X_{11,1}$ $X_{12,1}$	$X_{11,2}$ $X_{12,2}$
B_2	0	$X_{21,1}$ $X_{22,1}$ $X_{32,1}$	$X_{21,2}$ $X_{22,2}$ $X_{32,2}$

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 1. Suppose there are two positive and two negative bags each of which has two instances in the two-dimensional feature space shown in Figure 1a. 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 1b 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 2. Depending on the application, our proposal is flexible in generating average and maximum type of features which are famous in the MIL domain [12].



(a) Bag 1 and Bag 2 are negative labeled bags. Bag 3 and Bag 4 have positive labels. All bags and prototypes have two features.

(b) Average distances of bags to prototypes. A good prototype candidate separates negative and positive labeled bags.

Figure 1. Representation of the bags and prototypes.

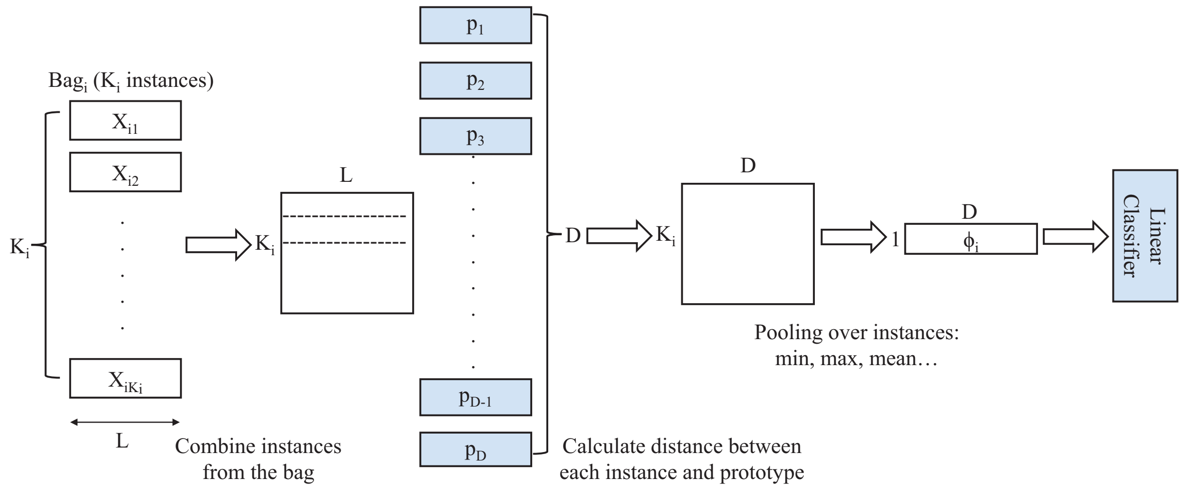


Figure 2. Overview of the architecture. Blue color shows the variables, which the model will be optimized over. L: Number of features in an instance, constant; D: Number of prototypes, hyperparameter; K_i: Number of instances bag i, varies between different bags.

As illustrated in Figure 2, 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 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, all distances between a prototype and each instance should be calculated. Euclidean distance of prototype d to instance X_{ij}, $Dist(X_{ij}, d)$, can be defined as in Equation 1:

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

3.2. Distance feature extraction

Just as in 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 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. Most basic and widely used pooling operators having these characteristics are min, mean and max operators [12]. 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.

3.3. Model formulation

A formulation for the discussed method is given in this section. The formulation for min features is demonstrated for simplicity, which could easily be generalized to any combination of max, min, and mean. The sigmoid function, which is defined in Equation 2, is used for binary classification.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

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 in Equation 3. β_0 is the intercept, β_d represents 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. Φ_{id} corresponds to the minimum distance of bag i to prototype d after layer normalization. Logistic regression minimizes a cross-entropy loss function defined in Equation 4.

$$\hat{Y}_i = \beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id} \tag{3}$$

$$\mathcal{L}_{ce}(Y,\hat{Y}) = -Y ln(\sigma(\hat{Y})) - (1-Y) ln(1-\sigma(\hat{Y})) \tag{4}$$

Y is the label, and $\hat{Y}$ is the prediction in Equation 4. We can decompose the objective function for each bag i and solve the nonlinear optimization problem using gradient descent. Decomposed cross-entropy loss will be denoted as $\mathcal{L}_{ce}(Y_i,\hat{Y}_i)$. λ_w is the regularization parameter for linear classifier weights, λ_p is the regularization parameter for prototype-to-prototype distances, and λ_d is the regularization parameter for the extracted distances in the formulation. Regularization of 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 5.

$$\min_{P,\beta} \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i,\hat{y}_i) + \lambda_w \left\|\beta\right\|_1 - \lambda_p \sum_{d=1}^{D}\sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{N}\sum_{d=1}^{D} \Phi_{id} \tag{5}$$

$$\Phi_{id} = \min_{X_{ik} \in X_i} Dist(X_{ik}, P_d) \tag{6}$$

$$\hat{y}_i = \sigma(\beta_0 + \sum_{d=1}^{D} \beta_d \Phi_{id}) \tag{7}$$

The objective function given in Equation 5 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.

3.4. 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 from [27, 28]. 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. Namely, each row in the transformed distance space is scaled to zero mean and unit variance as illustrated in Equations 8a and 8b. Layer normalization is important because of two reasons. The first reason is that it stabilizes the issues that could occur during optimization due to the scale of distance features. Secondly, layer normalization reduces the sensibility of the linear classifier to the scale of distances while keeping the relative distance information. As a side benefit, we observe that as argued in [27], it speeds up the convergence.

$$\mu_i = \frac{1}{D}\sum_{d=1}^{D}\Phi_{id}, \sigma_i = \sqrt{\frac{1}{D}(\Phi_i - \mu_i)^2} \tag{8a}$$

$$\Phi_i = \frac{\Phi_i - \mu_i}{\sigma_i} \tag{8b}$$

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.

We calculate the distance from each instance to the prototypes to extract distance features at each training step. Given these features, the model learns a classifier to predict the bag class. Details of the algorithm are given in Algorithm 1.

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 [3]. MNIST database is an image database. The database has 60,000 training and 10,000 test images of handwritten digits.

In this case, each instance is an image, and each bag consists of images. The task is finding 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 3a and 3b. In this application, we only used min as the aggregator for better interpretation. For instance, looking at Figure 3a, we see that the second prototype looks a lot like a 9, and the classifier found a negative coefficient for 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 3b.

5. Experimental results

We compare the performance of the aforementioned model to other well-known approaches in the literature. MIL literature has 68 common data sets that vary from molecular activity prediction to image annotation. Details of these data sets can be found in Table 4. Approaches are generally tested on these data sets. Our experiments on classical MIL benchmark data sets demonstrate that the proposed framework is an accurate and efficient classifier compared to the existing approaches.

Table 4. MIL data sets.

Name	**Instances**	**Min**	**Max**	**Features**	**Bags**	**+ Bags**	**-Bags**
Musk 1 ⊕	476	2	40	166	92	47	45
Musk 2 ⊕	6598	1	1044	166	102	39	63

Table 4. (Continued).

Name	**Instances**	**Min**	**Max**	**Features**	**Bags**	**+ Bags**	**-Bags**
Mutagenesis 1 ⊕	10486	28	88	7	188	125	63
Mutagenesis 2 ⊕	2132	26	86	7	42	13	29
Protein ⊕	26611	35	189	8	193	25	168
Elephant ⊖	1391	2	13	230	200	100	100
Fox ⊖	1302	1	13	230	200	100	100
Tiger ⊖	1220	2	13	230	200	100	100
Corel, African ⊖	7947	2	13	9	2000	100	1900
Corel, Antique ⊖	7947	2	13	9	2000	100	1900
Corel, Battleships ⊖	7947	2	13	9	2000	100	1900
Corel, Beach ⊖	7947	2	13	9	2000	100	1900
Corel, Buses ⊖	7947	2	13	9	2000	100	1900
Corel, Cars ⊖	7947	2	13	9	2000	100	1900
Corel, Desserts ⊖	7947	2	13	9	2000	100	1900
Corel, Dinosaurs ⊖	7947	2	13	9	2000	100	1900
Corel, Dogs ⊖	7947	2	13	9	2000	100	1900
Corel, Elephants ⊖	7947	2	13	9	2000	100	1900
Corel, Fashion ⊖	7947	2	13	9	2000	100	1900
Corel, Flowers ⊖	7947	2	13	9	2000	100	1900
Corel, Food ⊖	7947	2	13	9	2000	100	1900
Corel, Historical ⊖	7947	2	13	9	2000	100	1900
Corel, Horses ⊖	7947	2	13	9	2000	100	1900
Corel, Lizards ⊖	7947	2	13	9	2000	100	1900
Corel, Mountains ⊖	7947	2	13	9	2000	100	1900
Corel, Skiing ⊖	7947	2	13	9	2000	100	1900
Corel, Sunset ⊖	7947	2	13	9	2000	100	1900
Corel, Waterfalls ⊖	7947	2	13	9	2000	100	1900
UCSB Breast Cancer ⊖	2002	21	40	708	58	26	32
Newsgroups 1, alt.atheism ⊗	5443	22	76	200	100	50	50
N.g. 2, comp.graphics ⊗	3094	12	58	200	100	50	50
N.g. 3, comp.os.ms-windows.misc ⊗	5175	25	82	200	100	50	50
N.g. 4, comp.sys.ibm.pc.hardware ⊗	4827	19	74	200	100	50	50
N.g. 5, comp.sys.mac.hardware ⊗	4473	17	71	200	100	50	50
N.g. 6, comp.windows.x ⊗	3110	12	54	200	100	50	50
N.g. 7, misc.forsale ⊗	5306	29	84	200	100	50	50
N.g. 8, rec.autos ⊗	3458	15	39	200	100	50	50
N.g. 9, rec.motorcycles ⊗	4730	22	73	200	100	50	50
N.g. 10, rec.sport.baseball ⊗	3358	15	58	200	100	50	50
N.g. 11, rec.sport.hockey ⊗	1982	8	38	200	100	50	50
N.g. 12, sci.crypt ⊗	4284	20	71	200	100	50	50
N.g. 13, sci.electronics ⊗	3192	12	58	200	100	50	50

Table 4. (Continued).

Name	**Instances**	**Min**	**Max**	**Features**	**Bags**	**+ Bags**	**-Bags**
N.g. 14, sci.med ⊗	3045	11	54	200	100	50	50
N.g. 15, sci.space ⊗	3655	16	59	200	100	50	50
N.g. 16, soc.religion.christian ⊗	4677	21	71	200	100	50	50
N.g. 17, talk.politics.guns ⊗	3558	13	59	200	100	50	50
N.g. 18, talk.politics.mideast ⊗	3376	15	55	200	100	50	50
N.g. 19, talk.politics.misc ⊗	4788	21	75	200	100	50	50
N.g. 20, talk.religion.misc ⊗	4606	25	79	200	100	50	50
Web recommendation 1 ⊗	2212	4	131	5863	75	17	58
Web recommendation 2 ⊗	2212	5	200	6519	75	18	57
Web recommendation 3 ⊗	2212	5	200	6306	75	14	61
Web recommendation 4 ⊗	2291	4	200	6059	75	55	20
Web recommendation 5 ⊗	2546	5	200	6407	75	61	14
Web recommendation 6 ⊗	2462	4	200	6417	75	59	16
Web recommendation 7 ⊗	2400	4	200	6450	75	39	36
Web recommendation 8 ⊗	2183	4	200	5999	75	35	40
Web recommendation 9 ⊗	2321	5	200	6279	75	37	38
Birds, Brown creeper ⊘	10232	2	43	38	548	197	351
Birds, Chestnut-backed chickadee ⊘	10232	2	43	38	548	117	431
Birds, Dark-eyed junco ⊘	10232	2	43	38	548	20	528
Birds, Hammonds flycatcher ⊘	10232	2	43	38	548	103	445
Birds, Hermit thrush ⊘	10232	2	43	38	548	15	533
Birds, Hermit warbler ⊘	10232	2	43	38	548	63	485
Birds, Olive-sided flycatcher ⊘	10232	2	43	38	548	90	458
Birds, Pacific slope flycatcher ⊘	10232	2	43	38	548	165	383
Birds, Red-breasted nuthatch ⊘	10232	2	43	38	548	82	466
Birds, Swainsons thrush ⊘	10232	2	43	38	548	79	469
Birds, Varied thrush ⊘	10232	2	43	38	548	89	459
Birds, Western tanager ⊘	10232	2	43	38	548	46	502
Birds, Winter Wren ⊘	10232	2	43	38	548	109	439

⊕ molecular activity prediction, ⊖ image annotation, ⊗ text classification, ⊘ audio recording classification

We repeat stratified 10-fold cross-validation five times. Randomly generated cross-validation indices and results of the benchmark models are taken from [30]. Considered benchmarks are APR (axis-parallel rectangles) [11], CCE (constructive clustering based ensemble) [31], citation-KNN (citation k-nearest neighbor) [10], D_{maxmin}, $D_{meanmin}$, D_{minmin} (MIL with bag dissimilarities) [12] and MILES (multiple instance learning via embedded instance selection) [15], miFV (MIL based on the Fisher Vector representation) [32]. For all experiments, initial prototypes are generated randomly, and logistic regression is used as the default classifier. The model was implemented in PyTorch [33]. Adam optimizer from [29] was used. Parameters were tuned using Bayesian optimization [34]. Regularization parameters were searched between 0.00005 and 0.005. Boundaries of learning parameters were 0.00001 and 0.01. Number of prototypes were selected among $\{2, 4, 6, 8, 10\}$. We

---

Algorithm 1: Prototype learning algorithm.

Definitions:
λ_w: Regularization parameter for linear classifier weights, λ_p : Regularization parameter for prototype distances, λ_d : Regularization parameter for instance to prototype distances 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**
      Dist(i,d) $\leftarrow 1/K_i \sum_{i \in B_i} \sum_{j=1}^{K_i} Dist(X_{ij}, d)$;
    end foreach
    Calculate targets, AUC;
    if $AUC \geq BestAUC$ **then**
      BestAUC $\leftarrow$ AUC, counter $\leftarrow 0$;
    end if
    Loss $\leftarrow \sum_{i=1}^{N} \mathcal{L}_{ce}(y_i, \hat{y}_i) + \lambda_w \|\beta\|_1 + \lambda_p \sum_{d=1}^{D} \sum_{d'=1}^{D} Dist(P_d, P_{d'}) + \lambda_d \sum_{i=1}^{N} \sum_{d=1}^{D} \Phi_{id}$;
    Run Optimizer [29]**:** 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 if
    if *counter = 3* **then**
      break;
      counter++;
    end if
  end while
end foreach

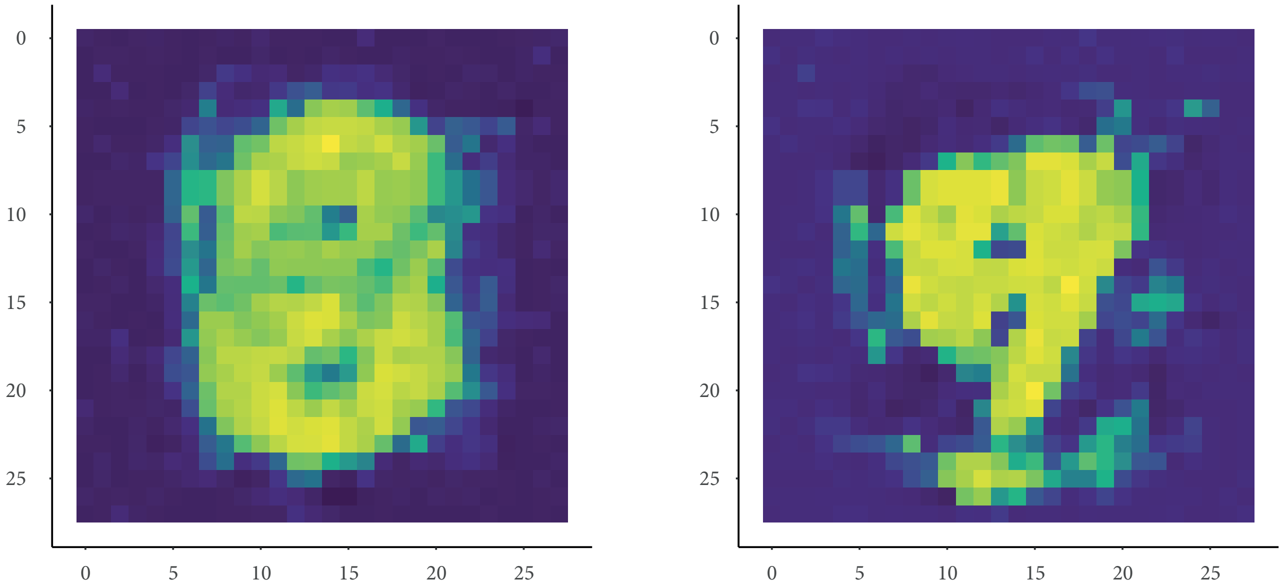
---

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 [27] to the distance features. Experiments were run on Windows Server 2016 operating system. The system has 10.0 GB installed memory (RAM) and a 2.30 GHz Intel Xeon(R) Processor.[1]

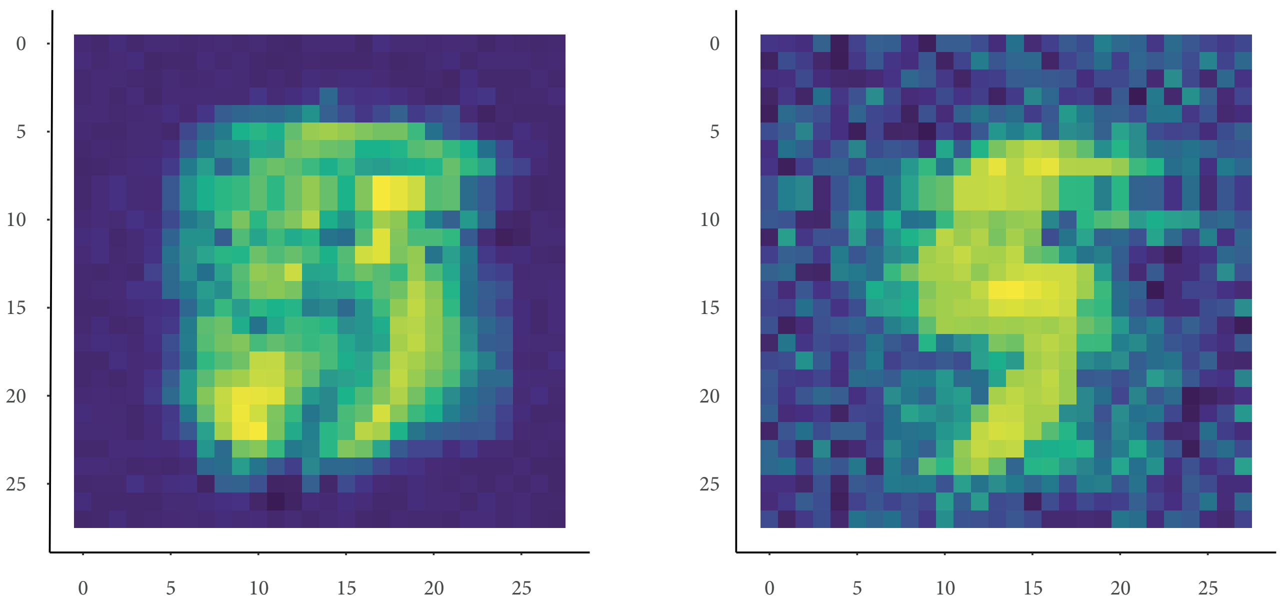
5.1. Classification accuracy

The solution approach in this study outperforms or at least does as well as all other well-known methods in terms of classification accuracy. Besides, this approach has much fewer parameters compared to a neural network. Area under the curve (AUC) is our primary performance measure to compare the approach with other well-known approaches. AUC is the area under the receiver operating characteristics (ROC) curve. The true positive rate is plotted against the false positive rate at a threshold parameter to create a ROC curve. AUC is used to show how successful the model makes classification, especially if there is a high imbalance between the numbers of positive and negative bags. Comparison of PL and other well-known approaches can be found in Table 5.

[1]Algorithm implementation and experiments can be found in the following link: https://github.com/mertyg/learning-prototypes



(a) Prototypes of finding the 9 problem. Classifier coefficients ofthe prototypes: Left:0.328, Right: -0.359.



(b) Prototypes of finding the 5 problem. Classifier coefficients ofthe prototypes: Left:0.265, Right: -0.359.

Figure 3. MNIST examples.

Table 5. Results of the algorithms.

Average of AUC	**APR**	**CCE**	**Cit.-KNN**	**Dmaxmin**	**Dmeanmin**	**Dminmin**	**MILES**	**miFV**	**PL**
Musk1	0.772	0.882	0.871	0.920	**0.945**	0.931	0.896	0.874	0.909
Musk2	0.806	0.781	0.850	0.956	**0.976**	0.956	0.823	0.786	0.892
Mutagenesis1	0.501	0.833	0.827	0.820	0.851	0.766	**0.910**	0.909	0.788
Mutagenesis2	0.458	0.727	0.688	0.590	0.647	0.345	**0.882**	0.867	0.873
Protein	0.509	0.643	0.552	0.561	0.523	**0.876**	0.872	0.873	0.867
Elephant	0.728	0.854	0.882	0.876	0.936	0.915	0.886	0.882	**0.936**
Fox	0.585	0.623	0.569	0.457	0.612	0.704	0.630	0.655	**0.737**
Tiger	0.583	0.832	0.752	0.761	0.853	0.850	0.826	0.860	**0.892**
CorelAfrican	0.580	0.840	0.862	0.939	**0.967**	0.966	0.836	0.851	0.902
CorelAntique	0.595	0.722	0.747	0.877	**0.922**	0.910	0.769	0.801	0.894
CorelBattleships	0.569	0.912	0.866	0.972	**0.981**	0.966	0.876	0.889	0.972
CorelBeach	0.596	0.969	0.926	0.972	0.983	**0.990**	0.977	0.980	0.979
CorelBuses	0.591	0.963	0.848	0.962	0.973	**0.981**	0.944	0.944	0.960
CorelCars	0.608	0.878	0.835	0.930	**0.948**	0.916	0.856	0.864	0.905
CorelDesserts	0.571	0.922	0.866	0.950	**0.974**	0.966	0.915	0.943	0.956
CorelDinosaurs	0.551	0.897	0.802	0.948	**0.983**	0.982	0.882	0.916	0.959
CorelDogs	0.570	0.805	0.752	0.894	**0.919**	0.911	0.838	0.856	0.844
CorelElephants	0.643	0.893	0.870	0.966	**0.983**	0.970	0.858	0.884	0.923
CorelFashion	0.868	0.950	0.909	0.983	0.990	**0.991**	0.904	0.928	0.953
CorelFlowers	0.636	0.844	0.833	0.932	0.947	**0.953**	0.856	0.890	0.885
CorelFood	0.860	0.985	0.934	0.993	**0.998**	0.996	0.979	0.984	0.983
CorelHistorical	0.773	0.978	0.942	0.984	**0.998**	0.994	0.985	0.987	0.929
CorelHorses	0.613	0.809	0.746	0.861	**0.920**	0.917	0.798	0.815	0.859
CorelLizards	0.553	0.938	0.897	0.960	**0.980**	0.971	0.940	0.934	0.953
CorelMountains	0.981	0.988	0.979	0.998	**1.000**	0.999	0.988	0.993	0.963
CorelSkiing	0.509	0.875	0.770	0.956	**0.960**	0.953	0.867	0.873	0.923
CorelSunset	0.495	0.728	0.706	0.803	**0.837**	0.751	0.704	0.751	0.766
CorelWaterfalls	0.595	0.836	0.865	0.962	**0.975**	0.966	0.839	0.896	0.891
UCSBBr.Cancer	0.569	0.644	0.706	0.725	0.831	0.791	0.823	0.848	**0.908**
Newsgroups1	0.500	0.792	0.803	0.905	**0.941**	0.500	0.667	0.654	0.837

Table 5. (Continued).

Average of AUC	**APR**	**CCE**	**Cit.-KNN**	**Dmaxmin**	**Dmeanmin**	**Dminmin**	**MILES**	**miFV**	**PL**
Newsgroups2	0.508	0.660	0.638	0.895	**0.898**	0.554	0.718	0.614	0.870
Newsgroups3	0.500	0.622	0.584	**0.818**	0.810	0.500	0.650	0.660	0.763
Newsgroups4	0.508	0.678	0.636	0.822	**0.857**	0.479	0.704	0.663	0.845
Newsgroups5	0.488	0.609	0.585	**0.853**	0.852	0.559	0.649	0.638	0.822
Newsgroups6	0.500	0.742	0.732	0.865	**0.890**	0.572	0.676	0.650	0.882
Newsgroups7	0.496	0.648	0.633	0.752	**0.790**	0.547	0.645	0.654	0.742
Newsgroups8	0.498	0.692	0.570	0.840	**0.870**	0.460	0.713	0.623	0.764
Newsgroups9	0.500	0.806	0.818	0.348	0.326	0.500	0.699	0.616	**0.881**
Newsgroups10	0.500	0.747	0.821	**0.918**	0.914	0.476	0.624	0.652	0.747
Newsgroups11	0.492	0.794	0.714	**0.968**	0.958	0.460	0.667	0.634	0.810
Newsgroups12	0.510	0.756	0.806	**0.868**	0.840	0.466	0.634	0.642	0.786
Newsgroups13	0.500	0.528	0.621	0.932	**0.946**	0.500	0.468	0.632	0.921
Newsgroups14	0.500	0.777	0.780	0.922	**0.942**	0.465	0.666	0.641	0.728
Newsgroups15	0.500	0.790	0.763	0.874	**0.905**	0.517	0.621	0.644	0.858
Newsgroups16	0.500	0.787	0.773	0.879	**0.898**	0.509	0.640	0.627	0.882
Newsgroups17	0.494	0.766	0.692	0.818	**0.874**	0.531	0.680	0.620	0.760
Newsgroups18	0.500	0.825	0.845	0.833	**0.874**	0.463	0.688	0.666	0.793
Newsgroups19	0.502	0.831	0.766	0.785	0.802	0.561	0.642	0.638	**0.856**
Newsgroups20	0.498	0.770	0.646	**0.839**	0.834	0.443	0.625	0.642	0.726
Web1	0.547	0.788	0.614	0.411	0.634	0.788	0.826	**0.834**	0.829
Web2	0.522	0.473	0.445	0.501	0.474	0.521	**0.700**	0.693	0.691
Web3	0.600	0.603	0.649	0.501	0.708	0.608	**0.772**	0.767	0.733
Web4	0.575	0.834	0.707	0.548	0.799	0.629	0.817	**0.847**	0.794
Web5	0.540	0.612	0.506	0.507	0.711	**0.794**	0.731	0.776	0.678
Web6	0.581	0.621	0.520	0.494	0.525	0.543	0.722	0.727	**0.741**
Web7	0.586	0.592	0.675	0.600	0.690	0.670	0.690	0.688	**0.744**
Web8	0.552	0.575	0.498	0.579	0.409	0.366	0.652	0.650	**0.710**
Web9	0.595	0.631	0.599	0.497	0.735	0.687	0.730	0.741	**0.767**
BrownCreeper	0.592	0.945	0.883	0.729	0.899	0.927	0.989	**0.992**	0.926
Ches.b.Chickadee	0.520	0.827	0.802	0.831	0.853	0.749	0.898	**0.910**	0.844

Table 5. (Continued).

Average of AUC	**APR**	**CCE**	**Cit.-KNN**	**Dmaxmin**	**Dmeanmin**	**Dminmin**	**MILES**	**miFV**	**PL**
Dark-eyedJunco	0.683	0.667	0.594	0.695	0.856	0.870	0.938	**0.947**	0.786
H.Flycatcher	0.534	0.992	0.884	0.718	0.944	0.883	**0.999**	0.999	0.996
HermitThrush	0.668	0.488	0.535	0.570	0.578	**0.892**	0.808	0.836	0.750
HermitWarbler	0.593	0.818	0.697	0.735	0.781	0.926	**0.981**	0.979	0.880
Olive-s.Flycatcher	0.644	0.878	0.794	0.853	0.896	0.924	**0.969**	0.963	0.933
Pac.lopeFlycatcher	0.531	0.836	0.699	0.723	0.754	0.769	0.957	**0.958**	0.831
Red-bre.Nuthatch	0.583	0.872	0.771	0.803	0.876	0.877	0.979	**0.982**	0.888
SwainsonsThrush	0.519	0.826	0.687	0.782	0.767	0.884	0.968	**0.981**	0.771
VariedThrush	0.584	0.869	0.782	0.751	0.840	0.940	**1.000**	**1.000**	0.949
WesternTanager	0.696	0.878	0.755	0.475	0.849	0.824	**0.988**	0.982	0.897
WinterWren	0.598	0.965	0.907	0.944	0.932	0.854	**0.993**	0.990	0.928
Performance Rank	9	7	8	4	1	6	5	3	2

We applied the procedure in [35] to compare the results of different approaches. Friedman test is a nonparametric test and concentrates on the average ranks. In our test, null hypothesis indicates that the average ranks of all different approaches are the same. The test concludes the average ranks performances between the approaches are significantly different at 5% alpha level (p-value $\cong 0$). Therefore, we continue with the Nemenyi test [36] to identify if a method outperforms the other in terms of average rank. If the average rank of the two approaches is greater than Nemenyi critical difference (CD) value, we can conclude that the performances of the two approaches are significantly different than each other. The best two performing approaches are $D_{meanmin}$ and PL in terms of average rank. The reported CD is 1.43 at 5% significance level. A scatter plot of AUC values of these two approaches can be found in Figure 4 for these two approaches. $D_{meanmin}$ outperforms PL in 46 data sets. However, averages of AUC values are closer to each other in both approaches except for a few data set such as Newsgroups10, Newsgroups11, and Newsgroups14. PL outperforms $D_{meanmin}$ in 25 data sets. $D_{meanmin}$'s average AUC values are high in Newsgroups data sets. However, it has a poor performance in Newsgroups 9. There is a significant difference between the two approaches in a few data sets including Web2, Web6, and Web8.

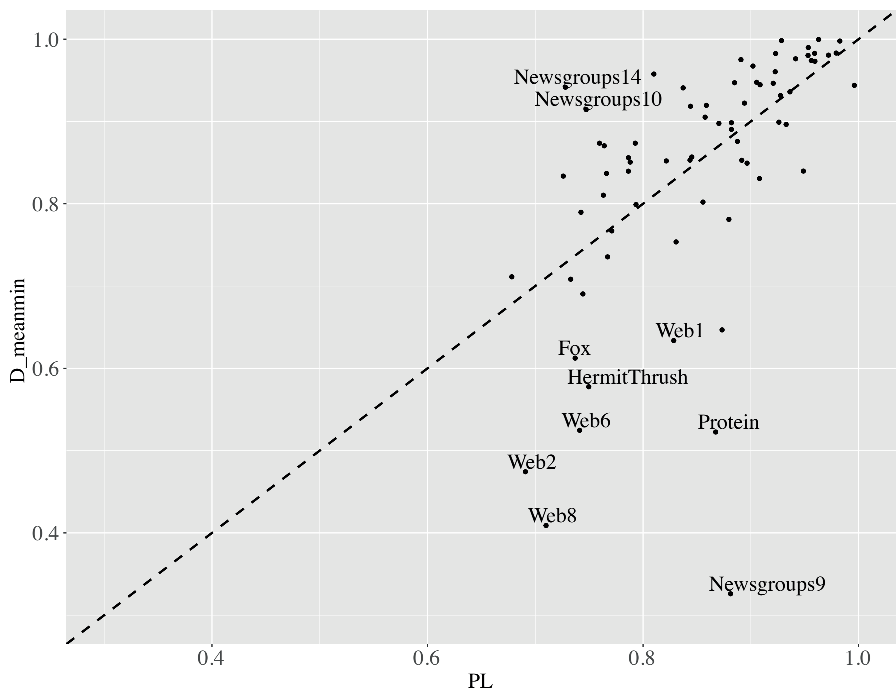


Figure 4. Comparison of AUC of $D_{meanmin}$ and PL.

5.2. Regularization

Experiments are repeated for Musk2 and CorelAntique data sets without prototype regularization to compare the algorithm performance. 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 PL outperforms the results without prototype regularization in terms of average AUC value. Apart from AUC performance, prototypes of PL without regularization have the value in a range of (–5, 5) whereas PL's prototypes have the value in a range of (–2.5, 2.5). This indicates that the regularization term prevents extreme values that may lead to overfitting.

5.3. Parameter sensitivity

Algorithm's performance changes under different parameter configuration. 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 5a and 5b. If learning rate is increased to 0.001 and above, average AUC suffers. Decreasing learning rates below 0.0001 does not increase performance even it brings high computational cost and slow convergence.

Experiments with number of prototypes $\in \{2, 4, 6, 8, 10\}$ can be found in Figure 5c. Best performance is obtained when the number of prototypes is 10. However, less number of prototypes (less than 6) shows poor performance in terms of classification power. The performance of PL is robust to the increasing number of prototypes due to the regularization applied to the weights (namely λ_w).

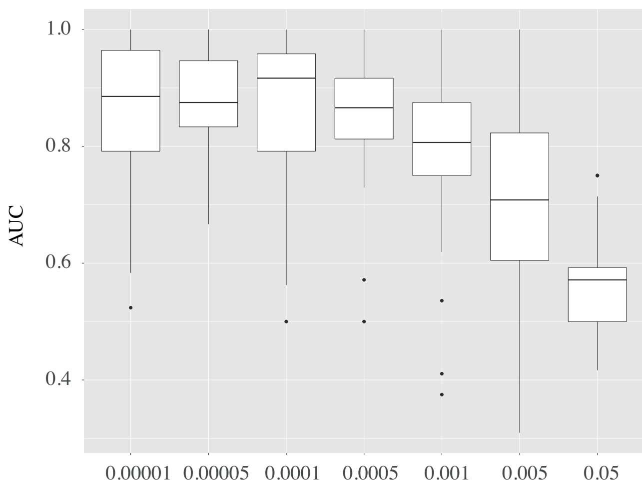
5.4. Time complexity

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. This is one of the highlights of prototype learning, which results in fast training and test times.

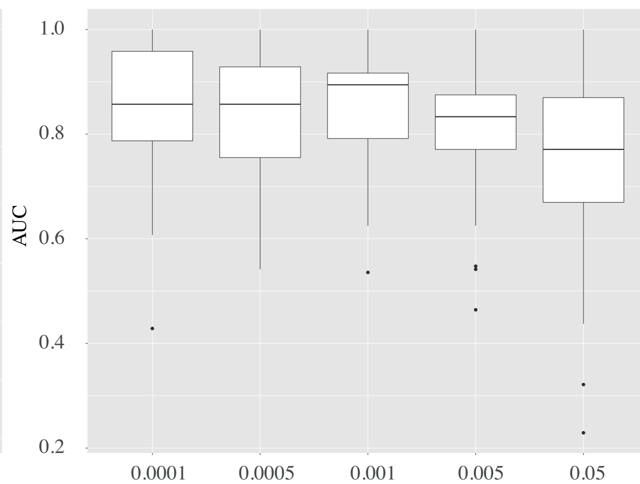
6. Conclusion

This work 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, 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. One can easily extend the same framework 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.

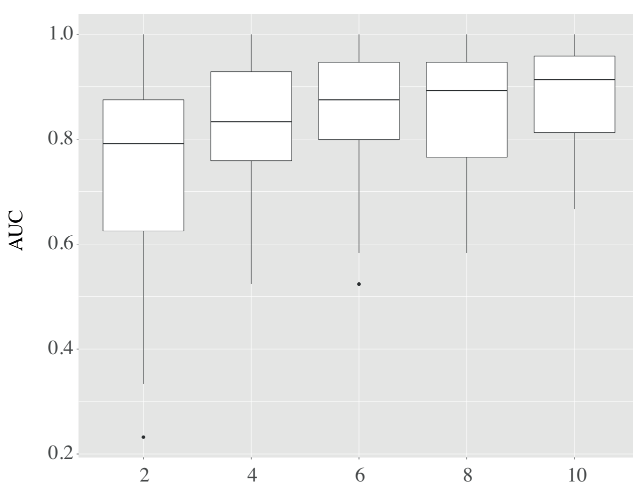
As future work, considering the flexibility of the architecture, one could incorporate more complex classifiers, different distance metrics, and different aggregation procedures to obtain more powerful models. We



(a) Learning rate of prototypes.



(b) Learning rate of weights.



(c) Number of prototypes.

Figure 5. Average AUC performance under different parameter configurations.

aim to present the simplicity of the approach. Therefore, logistic regression classifiers and Euclidean distance are adopted. Moreover, the algorithm is applied to the classical MIL benchmark data sets. The method's modification to different problems is an interesting research direction. A regression extension by changing the loss function is also another research direction.

Acknowledgments

This research is supported by the Air Force Office of Scientific Research Grant FA9550-17-1-0138.

References

[1] Al-Hussaini I, Xiao C, Westover MB, Sun J. SLEEPER: interpretable Sleep staging via Prototypes from Expert Rules. 2019.

[2] Simonyan K, Vedaldi A, Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. Workshop at International Conference on Learning Representations 2014.

[3] Ilse M, Tomczak JM, Welling M. Attention-based Deep Multiple Instance Learning. Proceedings of the 35th International Conference on Machine Learning 2018; 80: 2127-2136.

[4] Montavon G, Samek W, Müller KR. Methods for Interpreting and Understanding Deep Neural Networks. Digital Signal Processing 2018; 73:1-15. doi: 10.1016/j.dsp.2017.10.011

[5] Viola P, Platt J, Zhang C. Multiple Instance Boosting for Object Detection. Advances in neural information processing systems 18 2005: 1417-1424.

[6] Alpaydin E, Cheplygina V, Loog M, Tax D. Single- vs. Multiple-Instance Classification. Pattern Recognition 2015; 48 (9). doi: 10.1016/j.patcog.2015.04.006

[7] Raykar VC, Krishnapuram B, Bi J, Dundar M, Rao RB. Bayesian multiple instance learning: automatic feature selection and inductive transfer. In Proceedings of the 25th international conference on Machine learning (ICML '08) 2008. Association for Computing Machinery, New York, NY, USA, 808–815. doi: 10.1145/1390156.1390258

[8] Weidmann N, Frank E, Pfahringer B. A Two-level Learning Method for Generalized Multi-instance Problems. Machine Learning: ECML 2003: 468-479. doi: 10.1007/978-3-540-39857-8_42

[9] Li Y, Tax D, Duin R, Loog M. Multiple-instance learning as a classifier combining problem. Pattern Recognition 2013; 46 (3): 865–874. doi: 10.1016/j.patcog.2012.08.018

[10] Wang J, Zucker JD. Solving the multiple-instance problem: A lazy learning approach.Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000) 2000: 1119-1126.

[11] Dietterich T, Lathrop R, Lozano-Pérez T. Solving the Multiple Instance Problem with Axis-Parallel Rectangles. Artificial Intelligence 2001; 89 (1-2): 31-71. doi: 10.1016/S0004-3702(96)00034-3

[12] Cheplygina V, Tax D, Loog M. Multiple Instance Learning with Bag Dissimilarities. Pattern Recognition 2015; 48 (1): 264-275. 10.1016/j.patcog.2014.07.022.

[13] Tax D, Loog M, Duin R, Cheplygina V, Lee WJ. Bag Dissimilarities for Multiple Instance Learning. 7005. 222-234. doi: 10.1007/978-3-642-24471-1_16

[14] Pekalska E, Duin RPW. The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. Singapore: World Scientific 2005.

[15] Chen Y, Bi J, Wang J. MILES: Multiple-Instance Learning via Embedded instance Selection. IEEE transactions on pattern analysis and machine intelligence 2007; 28: 1931-1947. doi:10.1109/TPAMI.2006.248

[16] Cheplygina V, Tax D, Loog M. Combining Instance Information to Classify Bags. Multiple Classifier Systems, submitted 2013. doi: 10.1007/978-3-642-38067-9_2.

[17] Cheplygina V, Tax DMJ, Loog M. Dissimilarity-Based Ensembles for Multiple Instance Learning. IEEE Transactions on Neural Networks and Learning Systems 2016; 27 (6): 1379-1391. doi: 10.1109/TNNLS.2015.2424254

[18] Yang HM, Zhang XY, Yin F, Liu CL. Robust Classification with Convolutional Prototype Learning. 2018. 3474-3482. doi: 10.1109/CVPR.2018.00366.

[19] Snell J, Swersky K, Zemel R. Prototypical Networks for Few-shot Learning. Advances in Neural Information Processing Systems 2017; 30: 4077-4087.

[20] Wang X, Yan Y, Tang P, Bai X, and Liu W. Revisiting multiple instance neural networks: Pattern Recognition 2018; 74: 15–24. doi:10.1016/j.patcog.2017.08.026

[21] Wu J, Yu Y, Huang C, Yu K. Deep multiple instance learning for image classification and auto-annotation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015: 3460-3469. doi: 10.1109/CVPR.2015.7298968.

[22] Tang P, Wang X, Bai S, Shen W, Bai X et al. PCL: Proposal Cluster Learning for Weakly Supervised Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2020; 42 (1): 176–191. doi:10.1109/TPAMI.2018.2876304

[23] Fang W, Chang L, Wei K, Xiangyang J, Jianbin J et al. C-MIL: Continuation Multiple Instance Learning for Weakly Supervised Object Detection 2019. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/CVPR.2019.00230.

[24] Dennis DK, Pabbaraju C, Simhadri HV, Jain P. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18) 2018. Curran Associates Inc., Red Hook, NY, USA, 10976–10987.

[25] Angelidis S, Lapata M. Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis. Transactions of the Association for Computational Linguistics 2018; 6: 17-31. doi:10.1162/tacl_a_00002

[26] McFee B, Salamon J, Bello JP. Adaptive Pooling Operators for Weakly Labeled Sound Event Detection. IEEE/ACM Transactions on Audio, Speech and Language Processing 2018; 26 (11): 2180–2193.

[27] Ba JJ, Kiros RJ, Hinton GE. Layer Normalization. CoRR 2016. arXiv preprint arXiv:1607.06450

[28] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32Nd International Conference on International Conference on Machine Learning 2015; 37: 448-456.

[29] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations, ICLR 2015. https://dblp.org/rec/bib/journals/corr/KingmaB14

[30] Kucukasci E, Baydogan MG. Multiple Instance Learning Repository. 2018. http://www.multipleinstancelearning.com/

[31] Zhou ZH, Zhang ML. Solving multi-instance problems with classifier ensemble based on constructive clustering. Knowledge and Information Systems 2007; 11(2): 155-170. doi: 10.1007/s10115-006-0029-3

[32] Wei XS, Wu J, Zhou ZH. Scalable algorithms for multi-instance learning. IEEE transactions on neural networks and learning systems 2017; 28 (4): 975-987. doi: 10.1109/ICDM.2014.16

[33] Paszke A, Gross S, Chintala S, Chanan G, Yang A et. al. Automatic differentiation in PyTorch 2017.

[34] The GPyOpt authors. GPyOpt: A Bayesian Optimization framework in python. 2016. http://github.com/SheffieldML/GPyOpt/

[35] Friedman M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. The Annals of Mathematical Statistics 1940; 11 (1): 86-92.

[36] Nemenyi P. Distribution-free multiple comparisons. 1963.