IN4320 Machine Learning: Assignment 6
Multiple Instance Learning: Image Classification

Georgios Dimitropoulos: 4727657

June 20, 2018

1) The Naive MIL classifier

a)

No answer needed, just inspection of the given files.

b)

No answer needed, just reading of the images.

c)

In this question, we implemented the function **extractinstances** that segments an image using the Mean Shift algorithm (through using the supplied MATLAB function **immeanshift**), computes the average red, green and blue color per segment and returns the resulting features in a data matrix. Through experimentation, we set both the width parameters for apple and banana images for the **immeanshift** function to be 40 as we found that through these values the background and the foreground were segmented well in the first images.

d)

In this question, we created the function **gendatmilsival** that creates a MIL dataset, by going through all apple and banana images, extracting the instances per image and storing them in a Prtools dataset using the supplied MATLAB function **bags2dataset** and the resulting dataset contained all instances of all images and the label of each of the instances was copied from the bag label. We gave the apple and banana objects a class label of 1 and 2 respectively. We obtained 120 bags that represent the 120 images of apples and bananas. Every instance in the bags has 3 features which correspond to the average RGB value. The number of instances per bag varies in the range from 3 and 8. In Figure 1 the scatterplot of the instances from the two classes can be depicted. Every pair of features in each instance is plotted as a subscatterplot. The instances of apples are represented with blue color and the instances of the bananas are represented with red color. It can be observed that the instances of the bananas always reside in a some small area in contrast to the instances of the apples which have a more diffuse attitude.

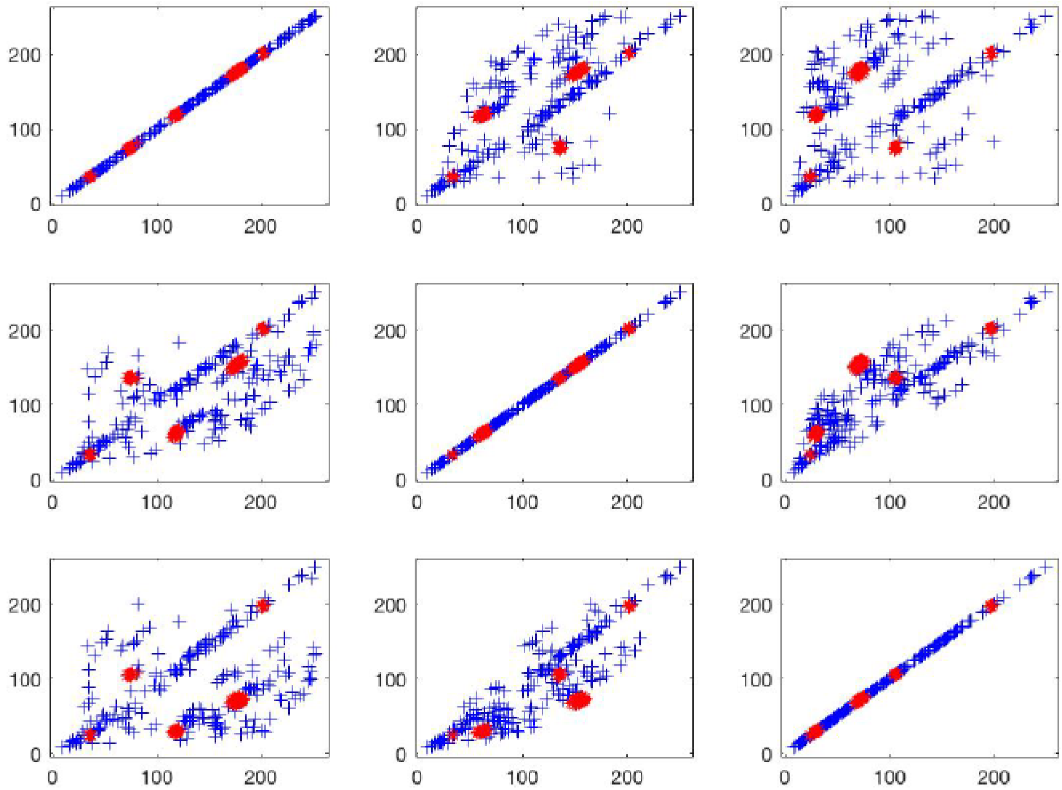

Figure 1: Scatterplot of the instances from the two classes

e)

In this question we created the function **combineinstlabels** that accepted a list of labels and gave as an output a single label which was obtained by majority voting.

f)

In this question, initially we trained the Fisher classifier. After that, we applied this trained classifier to each instance in a bag, classified the instances (using **labeled**), combined the label outputs (using **combineinstlabels**) and got a bag label. 33 apple images were misclassified to be banana and 10 banana images were misclassified to be apple. Hence, we had an error rate of 35.83%. However, based on the fact that the training and testing of the classifier was performed in the same dataset, we can conclude that this error estimate is not trustworthy. In order to be able to estimate the classification error in a trustworthy way, we split all the dataset into two disjoint sets randomly with 60 bags in each set (i.e. 30 apple bags and 30 banana bags), namely the train and the test set respectively. Also, the average error rate of 100 iterations was used in order to obtain more representative results and the corresponding error can be depicted in figure 2. Following this procedure, we found a mean of classification error of 44.31% and a standard deviation of the error 6.42%.

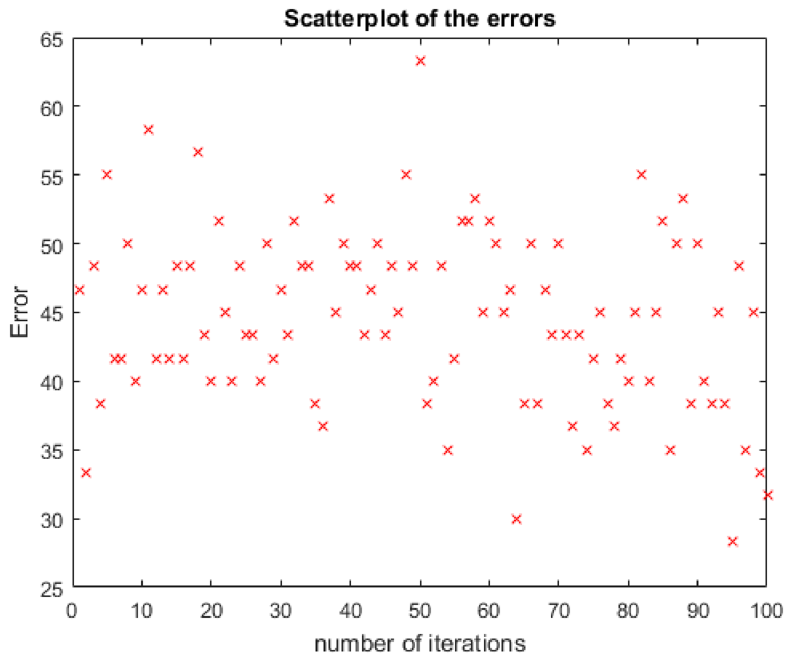

Figure 2: Error rate vs Number of Iterations

g)

In the previous procedure, the average RGB value was used as features in instances. In order to be able to improve the performance of this classifier, we could examine different values. For instance, the gradient of the RGB value, the gray scale value of the image, the covariance and different features could also be examined. Furthermore, a K-Means clustering in the segmentation procedure could also be examined. Finally, a different adjustment of the width parameter during the extraction of the instances may improve the performance of this classifier.

2) MILES

a)

In this question we implemented the function **bagembed** that represented a bag of instances B_i by a feature vector $m(B_i)$ through using the equation (7) from [1]. Again, in order to be able to estimate the classification error in a trustworthy way, we split all the dataset into two disjoint sets randomly. 30 images were chosen randomly from apples images and 30 from bananas images and thus we had a total of 60 images as train set and the remaining 60 images as test set. After the extraction of instances, we had 497 instances in the dataset. The feature vector $m(B_i)$ obtained had a length of 497. More specifically, the size of the $m(B_i)$ for our training and testing data was 60×497 for each of them.

b)

In this question we made a Prtools dataset with the vectors $m(B_i)$ and their corresponding labels y_i. In the sequel and after the aforementioned split of our dataset into train-test sets, we trained a L_1-support vector classifier through using the supplied MATLAB function **linkonc**.

c)

Here, we tested the LIKNON classifier on the test set in order to calculate the error rate. The average error rate of 100 iterations was used in order to obtain more representative results and the corresponding error can be depicted in figure 3. Following this procedure, we found a mean of classification error of 22.55% which is much less than the error rate of the Naive MIL classifier and a standard deviation of the error 4.58% . Thus, this classifier performs really better than the

naive MIL classifier. Also, in this classifier, in average, 10 apple images were misclassified to be bananas and 3 banana images were misclassified to be apples.
Finally, in order to improve the performance of MILES classifier, different features of instances could be examined such as the minimum and maximum of RGB or we could also increase the number of features in each instance. Furthermore, the search space is reduced significantly as the features are constrained to be derived from the training bags. If we relax this constraint, then the performance may be improved significantly. In addition, due to high storage limitations, the required storage space could be significantly reduced though using an instance similarity measure which produces sparse features. Finally, by optimizing the threshold in which the decision rule is based on, the false positive rate may be reduced.

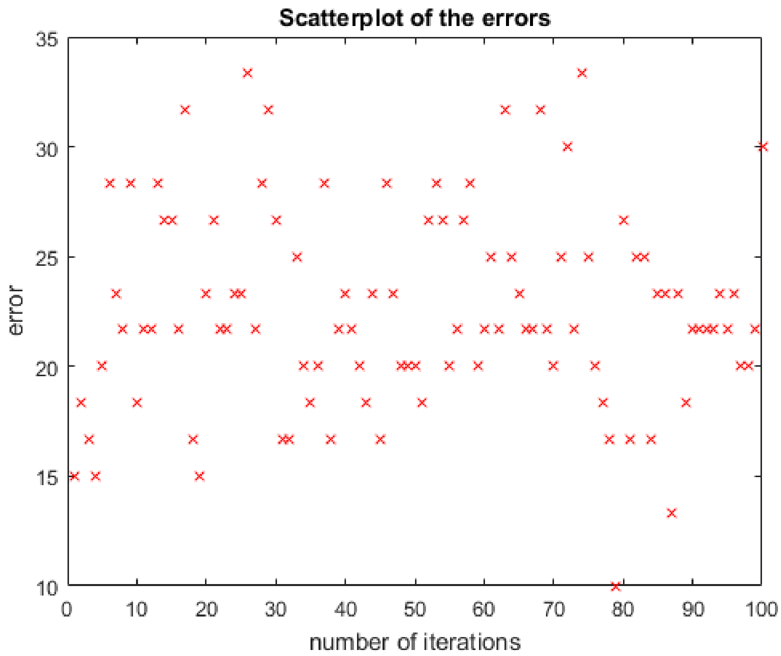

Figure 3: Error rate vs Number of Iterations

3) Another MIL classifier

a)

In this question we tried to make an implementation of our own MIL classifier through a function that we created and is named **AnotherMIL**. Our implementation consists of two steps. Namely we have:
Step 1) In this step, initially we extract the instances of all the apples and bananas objects and the features of the instances that are used are the minimum and maximum of the RGB value. In order to be able to do this we implemented a function named **minmaxextract** and as in the previous questions we also used in this case a width-parameter of 40, as through experimentations we found that through this value the background and the foreground were segmented well in the first images. Hence, we have 6 feature values for each instance. Through these extracted instances, labels are given in the apple and banana bags. In a similar manner, with the previous questions, in order to be able to estimate the classification error in a trustworthy way, we split all the dataset into two disjoint sets randomly. 30 images were chosen randomly from apples images and 30 from bananas images and thus we had a total of 60 images as train set and the remaining 60 images as test set.
Step 2) In the sequel, we computed a bag-distance matrix for all the bags in the train and test with the distance between bags to be defined as $D(B_i, B_j) = min_{k,l} \|x_{ik} - x_{jl}\|^2$, as described in the lecture slides. In order to be able to do this, we implemented two functions. Namely, the **distancebags** which computes the distance between two bags and the **dissimilaritybags** that calculates the distance of each pair of bags for a set of bags. After that we made our Prtools datasets for the train and test sets with the matrix D and the respective labels. Finally, we trained on the

train set the L_1-support vector classifier through using the supplied MATLAB function **linkonc** and evaluated its performance on the test set. The average error rate of 100 iterations was used in order to obtain more representative results and the corresponding error can be depicted in figure 4. Following this procedure, we found a mean of classification error of 39.28% and a standard deviation of the error 5.81%.

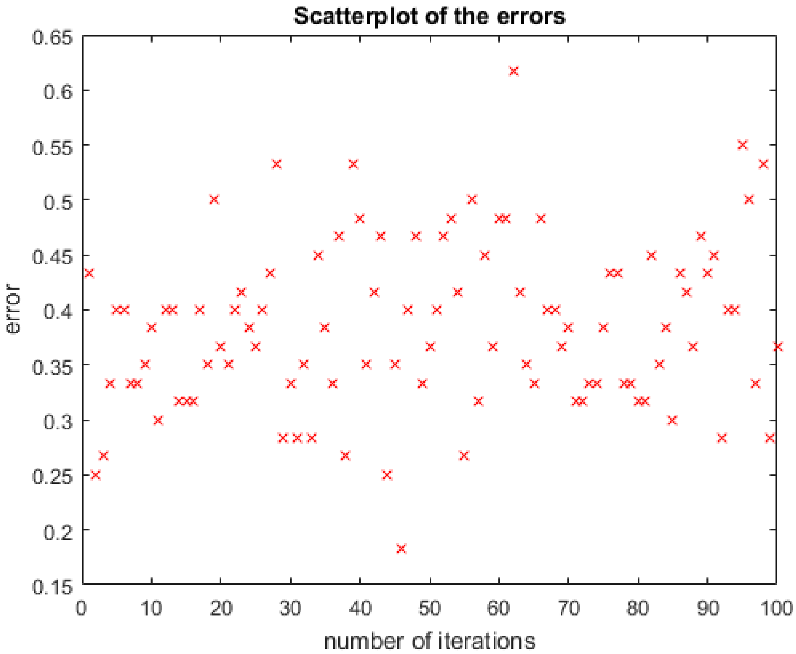

Figure 4: Error rate vs Number of Iterations

Finally, we compare the performance of this classifier with the Naive classifier and the MILES classifier. The error rate of every classifier can be observed in Figure 5. We can see that the MILES classifier achieves the best performance and that the classifier that we implemented in this question performs better than the Naive classifier.

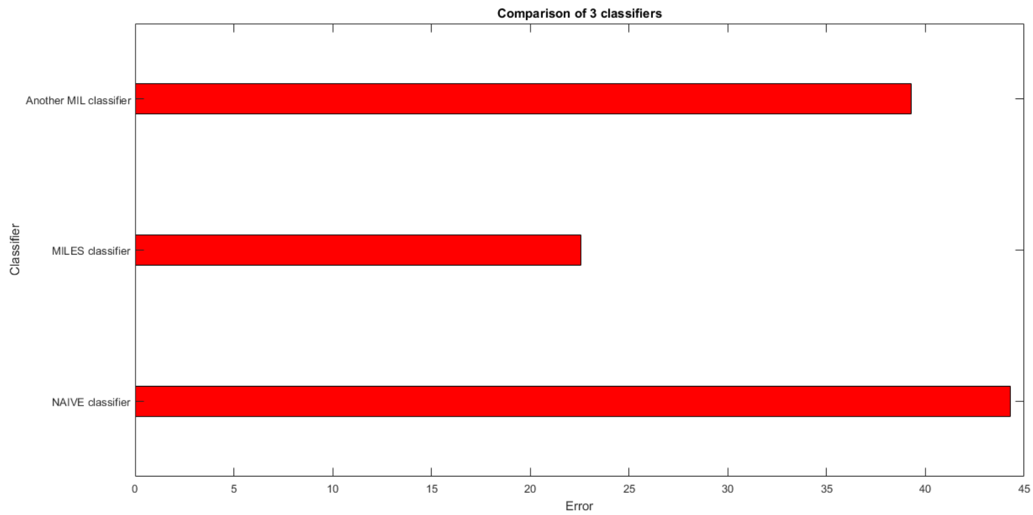

Figure 5: Comparison of 3 classifiers

Our MATLAB code for this question is as follows:

```
%Another MIL classifier
%Read images and extract instances
%{
cd 'sival_apple_banana/apple'
```

```
apples_files = dir('*.jpg');
num_apples = length(apples_files);
for i=1:num_apples
   currentfilename = apples_files(i).name;
   currentimage = imread(currentfilename);
   apples_images{i} = currentimage;
end
cd '../banana'
bananas_files = dir('*.jpg');
num_bananas = length(bananas_files);
for i=1:num_bananas
   currentfilename = bananas_files(i).name;
   currentimage = imread(currentfilename);
   bananas_images{i} = currentimage;
end
apples_width = 40;
bananas_width = 40;
cd '..'
cd '..'
[apple_lab, apple_bags] = minmaxextract(apples_images,apples_width);
[banana_lab, banana_bags] =extractinstances(bananas_images, bananas_width);
%}

load('data_another_classifier.mat')% load mat file for more efficient code
bags = [apple_bags, banana_bags];%compute bags
for epan=1:100%Perform 100 iteration for more representative results
% Random split of the dataset into train and test set
apples_random = randperm(60);
bananas_random = randperm(60)+ 60*ones(1,60);
train_set = {};
test_set = {};
% 60 train data, 30 apples and 30 bananas
for i = 1:30
    train_set{i} = bags{1, apples_random(i)};
    train_set{i+30} = bags{1, bananas_random(i)};
end
% 60 test data, 30 apples and 30 bananas
for i = 1:30
    test_set{i} = bags{1, apples_random(i+30)};
    test_set{i+30} = bags{1, bananas_random(i+30)};
end
% calculate the dissimilarities of the bags
bagdistance_train = dissimilaritybags(train_set);
bagdistance_test = dissimilaritybags(test_set);
% Prdataset
bagdis_train_dataset = prdataset(bagdistance_train, [ones(30,1); 2*ones(30,1)]);
bagdis_test_dataset = prdataset(bagdistance_test, [ones(30,1); 2*ones(30,1)]);
% train Liknon L1-support vector classifier
classifier = liknonc(bagdis_train_dataset, 30);
labels = labeld(bagdis_test_dataset, classifier);
% error rate
error_apples = 0;
error_bananas = 0;
for i = 1:30
```

```
    if labels(i) == 2
        error_apples = error_apples + 1; % misclassified apples
    end

    if labels(i+30) == 1
        error_bananas = error_bananas + 1; %misclassified bananas
    end
end
error_rate(epan) = (error_apples+error_bananas)/60;
end
disp(mean(error_rate)*100)
disp(std(error_rate)*100)
```

```
%Min-Max extract
function [label, bag_minmax] = minmaxextract(images,width)
size = length(images);
label = {};
for i = 1:size
   label = [label, im_meanshift(images{1,i},width)];
end
bag_minmax = {};
for i = 1:size
    image = images{1,i};
    num_instance = max(max(label{i}));
    instance = zeros(num_instance,6);
    for n = 1:num_instance
        instance_rmin = min(min(image(:,:,1).*uint8(label{i}==n)));
        instance_rmax = max(max(image(:,:,1).*uint8(label{i}==n)));
        instance_gmin = min(min(image(:,:,2).*uint8(label{i}==n)));
        instance_gmax = max(max(image(:,:,2).*uint8(label{i}==n)));
        instance_bmin = min(min(image(:,:,3).*uint8(label{i}==n)));
        instance_bmax = max(max(image(:,:,3).*uint8(label{i}==n)));

        instance(n,:) = [instance_rmin, instance_rmax, instance_gmin,
            instance_gmax, instance_bmin, instance_bmax];
    end
    bag_minmax = [bag_minmax, instance];
end
end
```

```
%Dissimilarity Bags
function [dissimilaritybags] = dissimilaritybags(bags)
size = length(bags);
for i=1:size
    for j=1:size
    dissimilaritybags(i,j)=0;
    end
end
for i = 1:size
    for j = 1:size
        dissimilaritybags(i,j) = distancebags(bags{i}, bags{j});
    end
end
end
```

```
%Distance Bags
function [distancebags] = distancebags(bag1, bag2)
[num_instance1, ~] = size(bag1);
[num_instance2, ~] = size(bag2);
```

```
for i=1:num_instance1
    for j=1:num_instance2
    distance(i,j)=0;
    end
end
for i = 1:num_instance1
    for j = 1:num_instance2
        distance(i,j) = norm(bag1(i,:)-bag2(j,:));
    end
end
distancebags = min(min(distance));
end
```

References

[1] *Yixin Chen , Jinbo Bi and James Z. Wang. MILES: Multiple-Instance Learning via Embedded Instance Selection. In IEEE Transactions on Pattern Analysis and Machine Intelligence, v.28 n.12, p.1931-1947, December 2006*